IDA

INSTITUTE FOR DEFENSE ANALYSES

Alignment of Army Integrated Core Data Model and Object Management Standards Category

Steven P. Wartik
Brian A. Haugh
Francisco L. Loaiza, Task Leader

Michael R. Hieb, IITRI

September 2001

IDA Paper P-3596

Log: H 01-000340

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
01-09-2001	Final rept.	xx-xx-2001 to xx-xx-2001

4. TITLE AND SUBTITLE
Alignment of Army Integrated Core Data Model and Object Management Standards Category
Unclassified

5a. CONTRACT NUMBER
DASW01-98-C-0067

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Haugh, Brian A. ;
Wartik, Steven P. ;
Loazia, Francisco L. ;
Hieb, Michael R. ;

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
Office of the Secretary of the Army
ODISC4 (SAIS-PAA-S)
2461 Eisenhower Avenue, Room 1126
Alexandria, VA22331-0700

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Interoperability between Command, Control, Communications, Computers, and Intelligence (C4I) and Modeling & Simulation (M&S) systems is an unrealized goal with great potential. Interoperability can facilitate training, simulation-based acquisition, mission planning and rehearsal, and course of action development and analysis. Recent interoperability research has concentrated on general-purpose approaches that can provide standards and reuse across a wide range of systems. One important component of interoperability is data model alignment, the degree to which the data models of two systems use the same elements. This paper presents a rigorous definition of data model alignment and uses it to assess the degree of alignment between the Army Integrated Core Data Model (AICDM) and the standard objects defined by the Object Management Standards Category (OMSC), two important emerging standards in the C41 and M&S communities, respectively. This assessment is used to make recommendations on changes to each model that would promote interoperability. The conclusion is that the OMSC standard objects need considerable work to model C4I data.

15. SUBJECT TERMS
Army Integrated Core Data Model (AICDM); Object Management Standards Category (OMSC); C4I; Interoperability; Data Model Alignment; Object Model; Modeling and Simulation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	189	Fenster-EM36, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

This work was conducted under contract DASW01 98 C 0067,Task BC-5-1943, for the Department of the Army and OASD (C3I). The publication of this IDA paper does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

INSTITUTE FOR DEFENSE ANALYSES

Alignment of Army Integrated Core Data Model and Object Management Standards Category

Steven P. Wartik
Brian A. Haugh
Francisco L. Loaiza, Task Leader

Michael R. Hieb, IITRI

Preface

This document was prepared under the task Order Object Oriented C2 Model for the Assistant Secretary of Defense for Command, Control, Communications, and Intelligence (C3I), in support of specific requirements identified by Department of the Army, Office of the Director of Information Systems for Command, Control, Communications, and Computers (ODISC4). It was performed in response to a task objective to develop a series of recommendations for how to align the underlying data representations that are required in C4I and in modeling and simulation (M&S) systems.

We wish to thank Major James Blalock, US Army, and Ms. Lesley Painchaud, IITRI, for their contributions to this effort and their review of this document.

The following Institute for Defense Analyses (IDA) research staff members were reviewers of this document: Dr. Richard J. Ivanetich, Dr. Reginald N. Meeson, Dr. Eugene Simaitis, Dr. Robert P. McDonald-Walker, and Mr. David A. Wheeler.

Contents

Figures

Tables

Executive Summary

Background

Interoperability between Command, Control, Communications, Computers, and Intelligence (C4I) and Modeling and Simulation (M&S) systems is one of the greatest challenges to the M&S community today. The Army needs simulations to train its forces, and to provide testing environments for new systems. But to use simulations with C4I systems, the Army has been developing point-to-point software interfaces.

In the past 10 years, M&S and C4I standards and development practices have diverged significantly. The two communities now use quite different data models as well as different architectures (M&S uses the High Level Architecture (HLA) while C4I uses the Defense Information Infrastructure Common Operating Environment (DII COE)). Data model compatibility is a fundamental issue. Major interoperability problems arise when there are data exchange requirements for C4I–M&S interoperation that are not supported by one side of the exchange. Similarly, interoperability problems occur if data representations differ significantly between systems, although these problems are not as bad as those caused by unsupported data exchange requirements.

It is desirable to have data compatibility to the fullest extent possible. General solutions to interoperability have not emerged to date, and lack of data compatibility appears to be a principal reason why. Beginning in 1993, with the third Army Tactical Command and Control Systems test, a "cottage industry" of custom, point-to-point C4I–M&S interfaces has grown up around the Army's family of Command and Control (C2) systems. Reuse, standardization, and interoperability were often not key design criteria, so most of these interfaces link a specific simulation to a specific C4I system and typically handle only a small subset of the messages or data the "target"—or stimulated—C4I system can accept and/or the simulation can pass.

Purpose

This report addresses the issue of developing common data models for Army C4I and M&S systems. It provides a base case study of compatibility (or extent of alignment) between existing C4I and M&S data models. This study serves the purpose of identifying compatibility (or alignment) problems which need to be resolved in order to enable development of common data models for C4I and M&S systems. In addition, it makes recommendations on addressing these problems in ways that move C4I and M&S modeling closer to the goal of common modeling standards.

Methodology

The study focuses on the principal formally sanctioned data standards in the Army C4I and M&S domains. In the C4I domain, this is the Army Integrated Core Data Model (AICDM) of the Office of the Director for Information Services for C4 (ODISC4). The AICDM is a relational model, and is the Army's data standard for tactical databases. In the M&S domain, the principal data standards are expressed in the standard objects of the Object Management Standards Category (OMSC) developed under the auspices of the Army Model and Simulation Office (AMSO). The OMSC consists of a set of standard objects, each of which presents an object-oriented model of some facet of simulation data and functionality. All of the currently developed OMSC standard objects (Unit, Platform, and Location) were analyzed.

The study performed an analysis of *alignment* to support an assessment of the potential compatibility and interoperability of systems based on the examined data standards. The report defines suitable technical concepts of alignment in order to enable quantitative assessments of alignment between these data models. Roughly speaking, these definitions declare that the AICDM and the OMSC are in alignment to the extent that AICDM modeling elements cover the data re-

quirements implicit in the OMSC standard objects.

The study assigned each modeling element a *degree of alignment,* the percentage of possible coverage. Ideally, each OMSC element ought to have a 100% degree of alignment with an AICDM element, meaning that these elements model the same data, and allowing an AICDM-based system and an OMSC-based system to interoperate with respect to these elements. But, if an OMSC element has no counterpart in the AICDM, there is 0% degree of alignment. Or the degree of alignment may be between 0% and 100%, as when the AICDM and the OMSC model similar types of data, but they do not match exactly. We expect that degrees of alignment lower than 100% entail significant amounts of effort to achieve interoperability.

Assessment of alignment between these two models is not straightforward. For one thing, the OMSC is a high-level model of architectural design. The OMSC standard objects are defined fairly informally relative to interoperability assessment needs. In any event, their definitions are incomplete. For another, the AICDM is a relational data model expressed in IDEF1X, whereas the OMSC uses object oriented models based on the Unified Modeling Language (UML). Finally, the models vary widely in scope and level of detail. The AICDM has approximately 370 entities in all views. The OMSC objects developed thus far have only 22 classes, where classes correspond in our alignment to entities.

As implied above, our analysis of alignment between the AICDM and the OMSC is unidirectional. It focuses on the extent to which AICDM modeling elements cover the data requirements implicit in the OMSC standard objects. It does not cover the extent to which OMSC standard objects can model the data that can be represented by AICDM modeling elements. We chose this direction for several reasons. First, it aids identification of any extensions to the AICDM that might be required to accommodate M&S data requirements (an objective of the task supporting this report). And, it is already known that most of the much greater number of AICDM data elements are not covered by the object models of the OMSC.

In some cases, assumptions about OMSC data elements are made where these seemed warranted by their (ambiguous) informal descriptions. These assumptions (e.g., data types, range of enumerated values) affect the alignment assessments in this report. Ambiguities and other problems in the OMSC standard leave open the possibility that simulation developers will interpret the OMSC standards in ways that lead to decreased degrees of alignment in actual implementations of the models. In addition, the chosen direction of alignment is not representative of the reverse direction (from the AICDM to the OMSC objects) which because of the AICDM's greater size is much less well aligned. Hence, an overall alignment assessment of these models (in both directions) would yield much lower levels of alignment.

Findings

The summary results of this alignment assessment are presented in Tables ES–1, ES–2, and ES–3. These summary results are based on the detailed analyses of alignment between individual entities and classes provided in this report.

The numbers in these tables are subjective, largely because of ambiguities in the OMSC specifications. The degree of alignment at the end of each table is an average; as such, it can be expected to contain some error. The IDA study team has calculated the standard deviation of the error. It is included in the tables. Thus, the value of 56% in Table ES–1 really means a 95% confidence level that the value is between 44% and 68%.

Given the ambiguities in the OMSC specifications it may be possible to enhance the alignment by adopting other interpretations of what the objects, classes and methods purport to represent and accomplish.

Analysis of these results clearly show that the Army has a serious problem with data model alignment between the C4I and M&S domains. Even if next generation training and testing simulations are built using standard simulation objects, developers will have to craft interfaces to transform data and, in many cases, create data that is missing from the other domain. The impact for acquisition of systems is significant. Millions of dollars will have to be spent after the systems are developed to interface incompatible models. These costs are avoidable to the extent that we can improve the degree of alignment of data prior to implementation.

Table ES–1. Summary of Unit Standard Object Degree of Alignment

OMSC Class	Degree of Alignment to the AICDM
Unit	51%
UnitGeometry	50%
Intel	37%
Communications	81%
SystemGroup	100%
Platform	55%
PlatformInfo	25%
Attrition	75%
Logistic	75%
Maintenance	42%
Supply	75%
C2	0%

Unit Standard Object Degree of Alignment: 56%
Standard deviation: 6%

Table ES–2. Summary of Platform Standard Object Degree of Alignment

OMSC Class	Degree of Alignment to the AICDM
Platform	62%
Sensor	30%
Weapon	50%
Movement	25%
Logistics	75%
Supply	55%
Maintenance	25%
Crew	100%
Communications	81%
Carrier	85%
PlatformFrame	25%
FrameComponent	25%

Platform Standard Object Degree of Alignment: 55%
Standard deviation: 6%

Table ES–3. Summary of Location Standard Object Degree of Alignment

OMSC Class	Degree of Alignment to the AICDM
Location	37%
Local	0%
LatLon	75%

Location Standard Object Degree of Alignment: 37%
Standard deviation: 4%

Recommendations

The recommendations from this study are in three classes: general recommendations, recommendations for the OMSC objects, and recommendations for the AICDM.

The general recommendation is that both the C4I and M&S communities work towards *a common data model* in the area of overlap between C4I and M&S requirements. The analysis shows that this is possible, but significant differences remain to be resolved that go beyond the current OMSC & AICDM models. This report found no theoretical reasons why C4I data cannot be represented in an M&S model or why M&S data cannot be represented in a C4I data model. C4I data models are more mature, so the primary responsibility currently falls upon the M&S community to move towards standard representations that, at the very least, can be aligned with C4I data. The M&S OMSC standards were insufficient in detail, insufficient in scope and did not structurally align to the AICDM. By comparison, only relatively minor changes to the AICDM are needed to accommodate M&S data requirements.

Recommendations for the OMSC: The report lists 34 specific recommendations to change the existing OMSC Standard Object classes. However, it also recommends that more sweeping revisions be made to correct structural misalignments and other problems documented in Appendix B, Critique of the OMSC Object Model.

Recommendations for the AICDM: The report lists 18 specific recommendations to change the AICDM. Many of these are already in data proposal packages submitted to the Defense Information Systems Agency (DISA), and are expected to be incorporated in future versions of that data model.

1. Introduction

Command, Control, Communications, Computers, and Intelligence (C4I) and Modeling and Simulation (M&S) are different communities with different backgrounds. There is currently little interoperability between C4I and M&S systems. It is not easy to embed C4I functionality into an M&S system, or vice versa. Yet interoperability would prove tremendously useful to both communities. The following examples are often cited as benefits of interoperability:

- Simulation-based acquisition (i.e., Requirements Development and Analysis, Testing, and Training)
- Development of doctrine and tactics techniques, and procedures
- Embedded training (both individual and collective)
- Course of action development and analysis
- Mission planning and rehearsal
- Execution monitoring

The Army spends around ten million dollars annually to achieve and maintain limited interoperability. It has implemented custom software interfaces linking specific M&S and C4I systems. These interfaces have not proven useful to other programs, which is not surprising because reuse, standardization, and interoperability were not part of the interfaces' design criteria. A general purpose approach to interoperability is one of the greatest challenges to the M&S community today.

1.1 Purpose of the Document

This report looks at a key area of C4I/M&S interoperability that has not been recognized to date. This area is data model alignment: the ability for C4I and M&S systems to share and exchange data based on a common model of the data each system manipulates. The premise of this report is simple. If data representation in M&S systems and C4I systems is very different, then interoperability will be difficult. This report provides a basis for assessing data model alignment, and makes recommendations for improving data model alignment when appropriate. This basis is termed *degree of alignment.* Degree of alignment is a quantitative measure of the interoperability between a C4I data model and an M&S data model.

The report assesses the degree of alignment between two data models:

- The Army Object Management Standards Category (OMSC) standard for Unit, Platform, and Location objects from the M&S domain.
- The Army Integrated Core Data Model (AICDM), the standard C4I data model developed for Army tactical databases.

The purpose of the assessment is to study the feasibility of using these two models as the basis for future interoperability work. These models are important emerging standards

that, it is hoped, will be used in the design and implementation of many systems. If the AICDM and the OMSC are aligned, future M&S and C4I system developers who based their systems on the AICDM or the OMSC will have a head start on interoperability. This report draws conclusions from the assessment on directions for these models to evolve that will promote interoperability.

1.2 Intended Audience

There are three intended audiences for this document:

- AICDM (and other C4I model) designers who want interoperability with M&S systems.
- OMSC (and other M&S model) designers who want interoperability with C4I data models.
- High-level DoD officials responsible for establishing directions for C4I and M&S systems.

This document assumes the reader is familiar with the IDEF1X notation for entity-relationship diagrams and the Unified Modeling Language (UML) notation for class hierarchies. [NIST 1993] and [Booch 1996] provide useful introductions and references to these notations.

1.3 Background

C4I/M&S interoperability is mainly accomplished by software *interfaces* established between specific systems. The development of C4I/M&S interfaces has not been one of the primary design requirements for either type of system. Most of the existing C4I interfaces to M&S have been developed as a separate component, added on after initial development. Existing interfaces typically handle a small subset of the messages or data necessary for interoperability, requiring significant human intervention to achieve realism for the training audience in an exercise. M&S systems, for instance, rarely handle free text messages or consider how a message is carried (i.e., communication effects). C4I systems have been subject to different design constraints than M&S systems, resulting in different standards, message formats and protocols. Since any interface between the systems must align these differences, the interface can become quite complex, and costly to develop and maintain.

Legacy systems and the lack of standard architectures have worked together in the past to frame the issue of C4I to M&S interoperability as one of interfaces. We believe that these interfaces, while necessary, are only one component of interoperability. Recent interface projects such as the Modular Reconfigurable C4I Interface (MRCI) [LSCZ 1998] provide lessons learned that have shaped our approach. *Complete interoperability can only be addressed by consideration of several different aspects such as standards, architectures, data models and processes.*

In recent years there has been some concerted effort to devise systematic approaches to interoperability. One of these approaches has yielded a framework that lays out several foundational areas in which progress must be made before interoperability can be

achieved [HS 2000]. This framework, shown in Figure 1, identifies the necessary components for a comprehensive solution:

- Architectures Alignment
- Common Data/Object Models
- Common Standards
- Processes to manage and align all efforts and respective results – and as a result of the above,
- Reusable Component Interfaces and shared solutions

The approach of the proposed *Architecture Alignment* recognizes that there are many ways to partition the "solution space". The C4I community has developed the Defense Information Infrastructure Common Operating Environment (DII COE) architectures. The simulation community has the High Level Architecture (HLA) [HLA 2000]. These architectures directly impact the technical basis upon which C4I and simulation systems are built. Alignment of architectures contrasts and resolves the differences in how architectures compartmentalize the "solution space" of the system(s) or system of systems.

Historically, the alignment of *Common Data Models* (C4I systems) with *Object Models* (simulation systems) is often ignored [HB 1999]. However, having M&S applications use the same or similar model representation as the C4I system with which they exchange data obviously minimizes translation. Without both model and architecture alignment, the efforts represented by the rest of the blocks comprising the "house diagram" of Figure 1 are limited to isolated interface successes such as seen today between stovepipe systems.

The development and use of *Common Standards* is another aspect of the alignment approach that must be worked into M&S system designs. Making sense of where and how to apply standards relies primarily on work being done on the architecture and data/object model alignment. Since little architecture and model alignment work has been done, it has often been difficult to set and use meaningful standards to assist interoperability challenges.

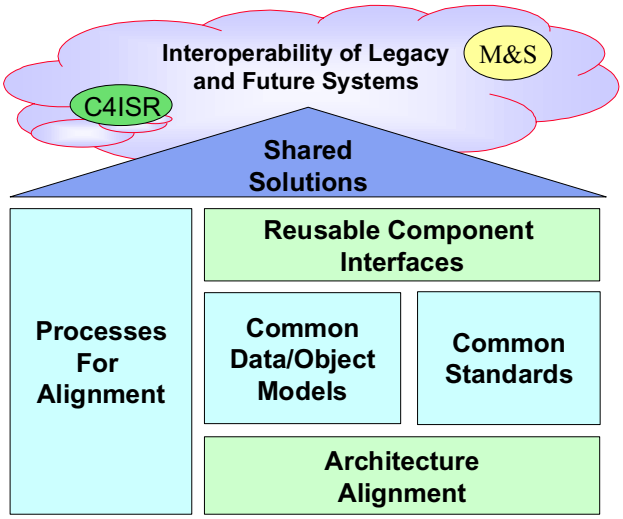


Figure 1. C4I/M&S Interoperability Components

Notice that the proposed approach sets the *Reusable Component Interfaces* on top of, and therefore dependent upon, the blocks below it. Compared to architectures, models and standards, the interfaces area has been a hotbed of activity. One answer to this apparent paradox is that interfaces can provide short-term solutions that are easier to envision and allow quicker successes in a world of disparate systems. Translators in these interfaces help to convert data between systems, but never really remove basic underlying incompatibilities of model representation or architecture misalignment in the original systems.

Finally, *Shared Solutions* between C4I and simulations—the roof of the house diagram in Figure 1—is supported by the work of all the blocks below it including the *Processes for Alignment*, which provides policy and procedures for evolving the other house blocks.

Efforts such as HLA have given those interested in interoperability the basis to pursue other areas. This paper focuses on the Common Data/Object Models box. It addresses *data alignment* between the two models—that is, knowledge of common data used in both M&S and C4I systems, along with the ability to share that data.

1.4 Data Model Alignment

The focus of this document is on data model alignment. Data model alignment is a key component of interoperability. Systems that use similar data models can share data easily. Suppose system A wants to incorporate functionality from system B. If their data models are aligned, system A can supply the data needed to invoke B's functions; also, system A can interpret the results of B's functions in terms of structures it possesses. But if their data models are not aligned, system A might lack all the necessary data to invoke one of B's functions. System A might have to manufacture the data by making limiting assumptions, thereby reducing the usefulness of interoperability with B. Or it might have to obtain the data from another source, such as another system (costly) or a human (slow). And if system A's data model does not account for all the data that B generates, system A must either be re-implemented or discard potentially valuable results.

Achieving data model alignment while ignoring the other components of Figure 1 will not achieve C4I/M&S interoperability. Each component has a necessary role. Moreover, even after achieving data model alignment, an organization can expect to devote considerable resources to achieving full interoperability. Nevertheless, achieving data model alignment is a significant step towards interoperability.

Currently, there is not much data model alignment between legacy C4I and M&S systems. Much of the disparity can be attributed to the different requirements and implementation technologies that drive each category of system. Many C4I systems are information systems. They have near real time, but not real time, performance constraints. They have taken advantage of this relative freedom to use commercial database management systems for data storage and data structures. This design decision is excellent from an economic standpoint. Whatever the drawbacks of commercial DBMSs—certainly no one has ever claimed their primary objective is to optimize run-time performance—they provide a powerful data modeling tool that can be used throughout software design and implementation.

M&S systems, as a rule, have less freedom. Their designers sometimes find that commercial DBMSs are too slow when applied to large data sets. Many M&S systems have met their performance requirements by implementing their own data models and data management subsystems.

1.5 Standard Data Models

Both camps have made efforts at standardization. Their motivation has not been interoperability between the C4I and M&S worlds so much as reuse within their own worlds. For example, all future Army Battle Command Systems (ABCS) will use a version of the Joint Common Data Base (JCDB) data model to promote interoperability of similar functions (see Section 2 for background on these models). Another example in the M&S world is the Army Materiel Systems Analysis Activity's development of Object Management Standards Category (OMSC), an object-oriented encapsulation of M&S functionality. The OMSC consists of a set of standard objects, each of which presents an object-oriented view of some facet of simulation data and functionality. The OMSC standard objects used in this report are discussed in more detail in Section 3.

Simulations have traditionally used highly specialized knowledge representations to achieve acceptable runtime performance. Standard representations have been slow to emerge due to the differing system components of simulations (software and hardware). The HLA addresses interoperability between simulations through specification of a Federation Object Model (FOM). The FOM allows diverse simulations to share certain classes of information during execution. However the FOM only is used for transferring data between models at run-time, and is not necessarily used when a system is being built. The purpose of the OMSC standard objects is to allow developers to use a common representation that saves development time and is a better representation.

Simulations use many different representations that are analogous to data models. In the case of common Army simulations, these representations in many instances, do not align with, or map to, the JCDB. If there is a mismatch between the simulation and C4I standards, then software translators will have to be built to align the data. Such translation software and associated interfaces are very costly and reduce interoperability through lack of functionality. In addition, if data elements are missing in simulations that are utilized in real world systems, interfaces become much more complex, as they must both create and synchronize such data.

The target data model in the Army C4I community is the Army Integrated Core Data Model (AICDM). Section 2 gives a detailed overview of the AICDM. Briefly, the AICDM is a relational data model based on several existing models, including the JCDB, the C2 Core Data Model, and the Army Land C2 Information Exchange Data Model (formerly called the Generic Hub Version 4, or GH4) developed to support multinational operations. The AICDM is the Army's target model for tactical databases. It is broader than the JCDB. The AICDM must accommodate both inter-service and international requirements, whereas the JCDB is more oriented towards ABCS systems, and more oriented towards the physical level. Although the two models are different, they use many of the same data entities and the degree of alignment determined in this paper would proba-

bly not differ substantially if the JCDB were used instead of the AICDM, at least with respect to the areas of C4I/M&S overlap.

One complicating issue in the analysis presented in this report is that the AICDM and the HLA use different paradigms. The C4I community is using relational data base modeling. The M&S community is primarily using object oriented modeling. There are no agreed-upon ways to compare the two. Rather than comparing an entity to an entity which would be possible if both standard models were based on the same relational paradigm, we must decide what to compare an entity to, which adds an additional layer to the analysis.

1.6 Assumptions

Several considerations have shaped this report. The first is that the OMSC standard object specifications do not contain sufficient detail to perform a full analysis of alignment. The AICDM provides specifications for its attributes giving data types and enumerated values. The OMSC specifications do not provide the equivalent information. Section 4 details how we have dealt with this problem, but it means that in many cases, we only provide recommendations for how to modify the AICDM. Due to the above-mentioned lack of detail in the OMSC specifications, the IDA study team had to assume a certain interpretation for the OMSC standard objects. Unfortunately, it is unlikely that the different users of the OMSC standard objects will all agree upon one interpretation, much less the one that has been adopted here.

A second assumption is one of scope. The AICDM has 377 entities, while the OMSC has 22 classes (the analog to relational data entities). Due to this large mismatch, only the OMSC classes have been compared to equivalent AICDM entities, and no attempt has been made to find matches in the OMSC for the numerous AICDM entities. This has the effect, as with the first assumption, of facilitating the alignment analysis over what it would be if we did not make either of these assumptions.

1.7 Organization of this Document

This document is organized as follows:

- Section 1 (this section) introduces the problems addressed by the report, the report's purpose and scope, and the report's intended audience.
- Sections 2 and 3 provide overviews of the AICDM and the OMSC models, respectively. The sections highlight facets of each model that relate to alignment. These sections are, however, only overviews. Readers seeking details should consult the references.
- Section 4 presents a precise definition of what alignment means in this report.
- Section 5 gives the process used to assess alignment. The process is intended to be repeatable and reusable. In other words, it can be applied as the AICDM and the OMSC evolve.
- Section 6 presents the results of applying the process to the OMSC model. Section 6 computes the degree to which the AICDM and the OMSC are currently aligned.

- Section 7 lists recommendations for changes to the AICDM and the OMSC, based on the results from Section 6.
- Appendix A provides details that underlie the results from Section 6.
- Appendix B gives a critique of the OMSC standard objects and the object model used to describe them.
- Appendix C lists the AICDM views extant at the time this report was written, and the status of each view.
- Appendix D presents an alternate categorization of recommendations in Section 7.
- This report analyzed the AICDM as of March 2000. Appendix E describes recommendations in this report that have been implemented in more recent versions of the AICDM.

2. Overview of the AICDM

2.1 Background

With the demise of the Soviet threat in 1989 and the concomitant budget reductions in the 90's, DoD realized the need to eliminate stovepipe systems and to take advantage of common hardware and software. This approach was extended to the data being collected and maintained by the Services and agencies, and DoD launched the standardization process under the DoD-8320.1 guidelines [DoD 1991].

The first model developed to capture the data requirements for the whole DoD enterprise, the DoD Enterprise Model, did not have a very robust C2 component. The Army proposed that the data model it had been developing for its multinational operations, namely the Generic Hub, be adopted and incorporated into the DoD enterprise data model (later renamed the DoD Data Model (DDM), and more recently the DoD Data Architecture (DDA)). The Defense Information Systems Agency (DISA) was tasked to perform the integration and the resulting structures constituted the initial version of the Command and Control Core Data Model (C2CDM).

From 1993–1997 the C2CDM continued to evolve and became the mandated model for all new C2 systems, both by the Joint Technical Architecture (JTA) [DoD 1999] and the Joint Technical Architecture – Army (JTA-A) [Army 2000]. It was also used as the point of departure for the JCDB, the model used to provide data interoperability to the ABCS.

However, the Army realized that many of the structures contained in the C2CDM were not keeping pace with its evolving data requirements, particularly with respect to the later versions of its C2 model for multinational operations, the Generic Hub, nor were the technical errors the C2CDM contained being corrected in a timely manner by DISA. As a result, the Army began a new effort to upgrade and expand the C2CDM, with a view to reflect these new data requirements as part of the DoD standardized model. The Army named this model the AICDM.

2.2 Central Entities of the AICDM

The AICDM models the battlefield using an Entity-Relationship (ER) model. It contains 377 entities. Ten of these are considered the key battlefield entities of the model, in that they model things most immediately recognizable and necessary to C4 operations: FEATURE, FACILITY, MATERIEL, ORGANIZATION and PERSON and their corresponding types FACILITY-TYPE, ORGANIZATION-TYPE, etc. These battlefield entities are defined as follows in the AICDM and the DDA:

- FEATURE: A set of characteristics, structures, or other entities that are of military significance.

- FACILITY: Real property, having a specified use, that is built or maintained by people.
- MATERIEL: An object of interest that is non-human, mobile, and physical.
- ORGANIZATION: An administrative structure with a mission.
- PERSON: A human being.

The idea behind this approach is that for each of the first five key entities, there is an associated entity that describes the class of the key entity. For example, there is an entity PERSON-TYPE associated with PERSON via an "is the type for" relationship. Where PERSON models an individual, PERSON-TYPE models characteristics of a class of persons, such as the rank of military personnel.

The AICDM also permits the specification of objectives, activities, goals, etc., and uses either the instances of the key entities or of their types to accomplish these functions, e.g., PLAN, SITUATION, TASK, TARGET, GUIDANCE. Many of the basic relationships between these activity related entities and the battlefield object types and object items are illustrated by the IDEF1X diagram schema in Figure 2.

The individual relationships among the battlefield entities (FACILITY, FEATURE, MATERIEL, ORGANIZATION, and PERSON) and their types are schematized to simplify the diagram (in other words, Object Types, Object Items, ESTABLISHMENT, HOLDING-ESTIMATE, and ASSOCIATION are notional). The more specific relationships between these entities which are relevant to our analysis appear in a series of IDEF1X diagrams in Appendix A. All of

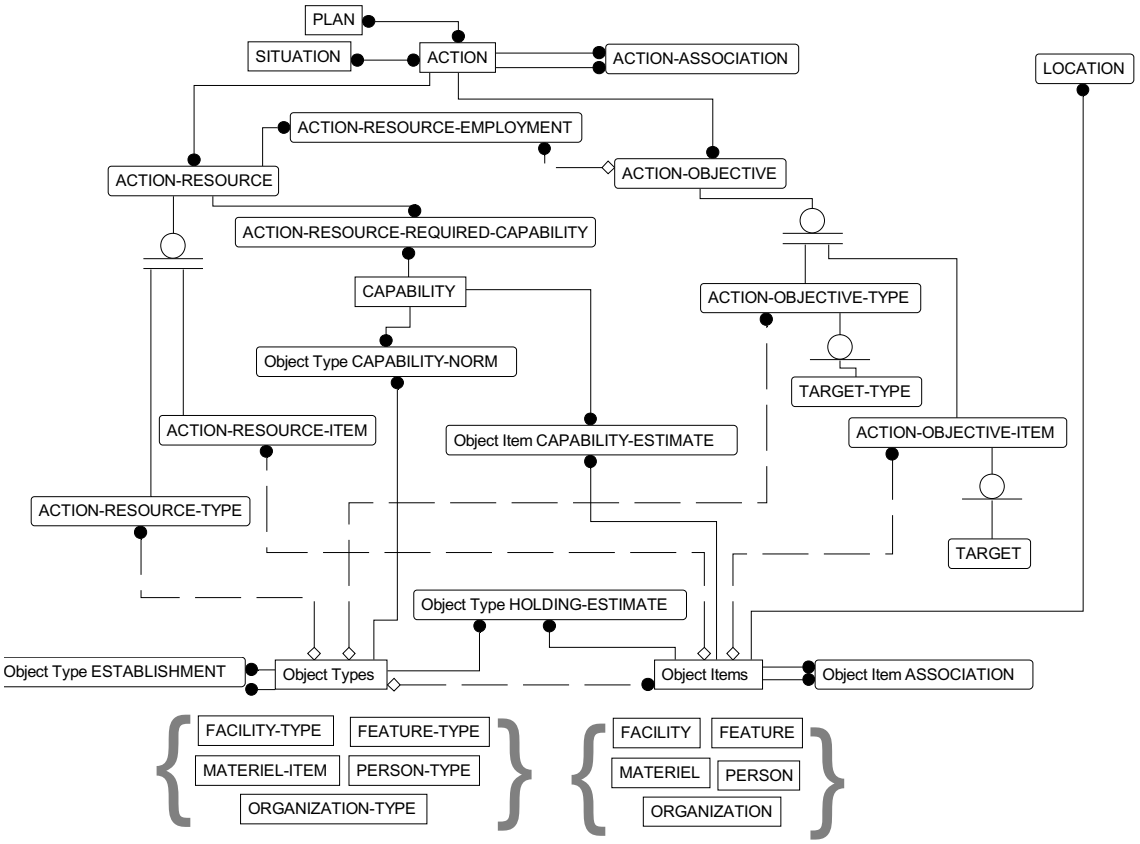


Figure 2. AICDM Core Relationships

these diagrams suppress the display of the attributes of AICDM entities to promote comprehension of the entity relationships. For details of the attributes and their metadata, see the full AICDM model, or the DDA.

2.3 Objectives of the AICDM

One of the main goals in creating the AICDM was to use it as a vehicle for bringing within the DoD standardization process all the new data requirements identified in the JCDB. Arguably, JCDB's intended widespread use in the C4I community could be viewed as a de facto data standard for information exchange among the community's members. The problem with this is other functional areas that at present neither interact with C4I systems nor use the JCDB as their reference model for information exchange. These functional areas may develop duplicative data structures on their own that will later necessitate translators to interface with present and future C4I systems, thereby defeating the overarching goal of the DoD data standardization program. As DISA moves to a faster and more responsive way of reviewing and approving the proposed changes to JCDB, it is likely that implementers using the JCDB will feel more comfortable with the idea of making the submittal of their new data structures an integral part of their work. At present the Army continues to endorse the need for using the DoD 8320 process as the way to achieve data interoperability in all its information systems.

The AICDM is meant to be a superset of the C2CDM. It incorporates data requirements identified in the various versions of the Generic Hub, as well as in other data models such as the Modernized Intelligence Database (MIDB).

The structures that constitute the AICDM are to become part of the DDA. They are created and approved in full compliance with the DoD 8320 procedures.

The AICDM is meant to be a logical data model. Its purpose is to capture data requirements and specify them to a sufficient level of detail to enable functional experts to assess the appropriateness of the proposed structures. Implementers remain free to adapt the model for its various physical implementations.

The AICDM constitutes a target model, and as such is meant to remain flexible and continue evolving. One of the advantages of maintaining the AICDM (as opposed to adopting the JCDB) as the data model for information exchange is that the AICDM work is not constrained by any delivery schedules. Additionally, because the structures of the AICDM are submitted to the DoD 8320.1-M-1 approval process (see Section 2.4), they receive inputs from all the relevant functional areas. The structures of the JCDB are only reviewed by those organizations which either use one of the current ABCS systems or have some need to exchange data with them. Functional areas outside of this community do not necessarily have visibility into the process and may not be aware of the solutions generated there. As a result they may neither be able nor inclined to adopt them and quite likely will generate their own, creating a potential for disconnects between their systems and those using the JCDB. The AICDM therefore provides a vehicle to avoid creating yet another Army stovepipe system, however broad this stovepipe may be, so that other Ser-

vices may take advantage of the work done within the Army, and thereby reduce the risk of decreased data interoperability.

2.4 The AICDM and DoD 8320

From the beginning, DoD has conducted data standardization efforts in accordance with DoD-8320.1-M-1 [DoD 1998] guidelines. According to DoD-8320.1-M-1, every entity and its attributes must be modeled using the IDEF1X methodology [NIST 1993]. Furthermore, DoD-8320.1-M-1 prescribes a set of metadata items required for every entity and attribute. Several of the metadata items were particularly important in the alignment analysis. Table 1, condensed from [DoD 1998], shows these items.

Table 1. DoD-8320.1-M-1 Metadata Items Used in Alignment Analysis

	Name	Definition
Entity Metadata	Entity Name	The label of an entity; must be a noun or noun phrase with the entire phrase connected by hyphens; must accurately reflect the characteristics (attributes) of itself, especially its domain.
	Definition Text	The narrative description of what an entity is.
Attribute Metadata	Standard Data Element Name	The label of an attribute, comprised of a minimum of an entity and generic element; may contain property modifier(s) providing additional descriptions; may utilize generic data; must be a noun or noun phrase and accurately reflect the characteristics (meta-data) of the attribute, especially domains.
	Data Type Name	The name of the way domain values are stored in a database. The generic data elements with class words having a data type of "integer" will be modified with a comment (comment text field) as follows: Data element using the data type "integer" should fit into a 32 bit representation. The high range value of a signed integer is limited to "2.1 billion" (in the range -2^{31} to $2^{31}-1$); data requirements of greater values should use the data types "floating point" or "fixed point".
	Domain Value Identifier	The actual codes that provide access to lists of categories of objects.

One of the requirements of the DoD-8320.1-M-1 process is that the data structures modeled using the IDEF1X methodology be in "third normal form". Third normal form involves the separation of data into bins that provide a "tight fit" to the data needed to be captured in the tables of a relational database. The downside of this is the proliferation of entities, and later on tables in the physical data base, which may require the DBMS to search in various tables, rather than a single one. Some DBMSs experience large performance degradation when the number of tables that need to be searched to extract the data becomes large. During physical implementation many of these tables are routinely combined into a single one and validation rules are implemented to ensure that data which at the logical level was depicted as a separate entity, and would have been in a separate table of the relational database, is placed in the appropriate portions of the "denormalized" table. With ever-increasing CPU speeds and overall better DBMS performance, the need to use this approach when generating the physical schema out of a logical

data model may become less necessary, and the logical and physical schemas of the C4I systems may remain fairly close to each other

2.5 Current Status of the AICDM

As indicated above, the AICDM is a planned expansion of current DoD C2 data model specifications in support of newly identified data requirements. The AICDM is an outgrowth of DoD's C2 Core Data Model (C2CDM), with important influences from the fourth version of the Generic Hub data model (GH4) and the Joint Common Database (JCDB). Begun in 1997, AICDM is an ongoing effort that will support force-level C2, including a common picture of the battlefield, information exchange (both pushed and pulled), and distributed collaborative planning, as well as integration with all the other functional areas, such as Intelligence, Air Operations, and Naval Warfare. AICDM is the Army's target model for tactical databases.

The AICDM is defined using entity-relationship (ER) modeling concepts. Because of limitations imposed by DISA, as well as the manpower available to generate the data proposal packages, the data structures of the AICDM become part of the DoD standard in finite increments. The portions of the AICDM that have gone through the DoD-8320.1-M-1 process and reached approval status are captured in so-called "views", i.e., extracts from the totality of the model[1]. Each view depicts a facet of a battlefield C4I model; the union of all the views is the complete AICDM model. Table 2 lists some of the approved AICDM views. Appendix C lists the complete set of the AICDM views and their status as of this time. These views give a flavor of the AICDM's intent: to model battlefield objects. The AICDM attempts to model anything that might be relevant to the warfighter. Personnel, materiel, organizations, terrain, and facilities are all within its purview.

Table 2. Selected AICDM Views and Their Purposes

View	Purpose
ACTION	A model of how an organization may describe the actions it undertakes, the objectives of those actions, and the resources needed to perform them.
CAPABILITY	A model of the expected and actual capabilities of a person, organization, or feature
Feature View	A model of something—physical or logical—that is of military significance.
ORGANIZATION	A model of administrative structures, with specific missions and their relationships.
Plan View	A model of how to achieve some objective over time.

It is important to understand that there is nothing special about the views. The full model is the ultimate definition of the AICDM. The views are derived from the full model. The set of views that the AICDM contained when the IDA team examined it initially for the purpose of this analysis had been created to support the data proposal packages being submitted to DISA by the Army Data Management Group at Ft. Belvoir to standardize the entities and attributes contained therein. In some cases the existing views constituted

[1] The Army maintains the AICDM views and specifications at the Army Operational Data Repository website, the URL of which is http://aodr-arch-odisc4.army.mil/.

excellent points of departure for the alignment analysis study. In other cases the IDA team had to create new ones from the pool of all AICDM entities to capture the corresponding classes of the OMSC. Such cases are noted in the text.

It is also important to understand that this report uses the term "view" casually. Technically, a view includes all the entities connected to those one would like to think of as making up the view. This is needed to ensure that no foreign keys appear in any of the tables without their parent entities being properly defined in the model. For instance, the AICDM's ORGANIZATION view contains an entity ORGANIZATION-FACILITY, which has an attribute FACILITY IDENTIFIER; yet the ORGANIZATION view does not include the FACILITY entity. According to the usual formal definition of a view, the ORGANIZATION view lacks *referential completeness*. However, a view with referential completeness quite often includes tens of entities that add little actual information. Data modelers customarily omit entities from a view that are not related to those relevant to the subject of the view either directly (i.e., they have either identifying or non-identifying relationships to the relevant entities in question), or form part of an associative pair with the focal entity of the view (e.g., FACILITY is excluded from the ORGANIZATION view even though the ORGANIZATION view includes the entity ORGANIZATION-FACILITY, which is a child of FACILITY).

2.6 AICDM Conventions

The AICDM has some conventions that bear mention. Knowledge of these conventions is necessary to understand certain material in this report. Some of these conventions are commonly used throughout the data modeling community. Others are specific to the AICDM in that they reflect specifics of the DoD 8320 guidelines.

2.6.1 Naming Conventions

AICDM entities and attributes follow simple and useful naming conventions. An entity's name is in upper case. If it is composed of multiple words, those words are separated by hyphens. PERSON and MATERIEL-ITEM are examples of entity names.

The attribute's name includes the name of its entity as a prefix. The entity PERSON has attributes PERSON IDENTIFIER, PERSON BIRTH DATE, etc. The attributes of the entity MATERIEL-ITEM include MATERIEL-ITEM NAME and MATERIEL-ITEM TYPE CODE. Note also the space between the entity name and the remainder of the attribute's name. This convention lets the reader determine an attribute's entity from the attribute's name. It also promotes brevity, since one can write "PERSON BIRTH DATE" rather than "the BIRTH DATE attribute of the PERSON entity" without being ambiguous.

Because the AICDM uses the IDEF1X methodology, a child entity inherits the key attributes of its parent entities; as a result, an AICDM entity such as PERSON may show an attribute named RACE CODE, coming from the entity RACE. In such cases, this report explicitly identifies the entity with which the attribute is associated.

2.6.2 Modeling Many-to-Many Relationships

Usually, a relationship in an ER model can be implemented in a relational database by the straightforward means of expressing it as a single relation, with each of the key attributes of the parent entity being inherited by the child entity. If two entities have a many-to-many relationship (consider PERSON and ORGANIZATION), the usual technique to implement that relationship is to create an intermediate associative entity that is the child of both entities related by the many-to-many relationship. This associative entity:

- Contains as attributes the keys of both the entities it relates. For example, the associative entity implementing the relationship between PERSON and ORGANIZATION, namely PERSON-ORGANIZATION, contains ORGANIZATION IDENTIFIER and PERSON IDENTIFIER.
- May contain attributes of its own that participate in the key, to permit the association of the same pair of instances of the key attributes from the parent entities if the role of the association is different or reflects the history of the possible values. For example, a person may serve in a unit at one time, leave for a couple of years, then come back to the same unit and serve again. Thus the associative entity PERSON-ORGANIZATION contains attributes to record the time a person begins each association with an organization.
- May contain non-key attributes of its own. For example, a person is associated with an organization for a defined time period. Thus the associative entity PERSON-ORGANIZATION contains attributes to record the time a person ends an association with an organization.

In the AICDM all many-to-many relationships are resolved via associative entities. See Figure 3.

The AICDM names the intermediate entity by combining the names of the two entities from the many-to-many relationship. It therefore has an entity PERSON-ORGANIZATION. The attributes of PERSON-ORGANIZATION follow the naming conventions in Section 2.6.1.

2.7 Instances, Types, and Quantities

When modeling one's own forces, it is advantageous to note individual persons and materiel insofar as is possible. In other words, the model should be capable of recording the names of all individuals in an organization and the serial numbers of all equipment the organization possesses.

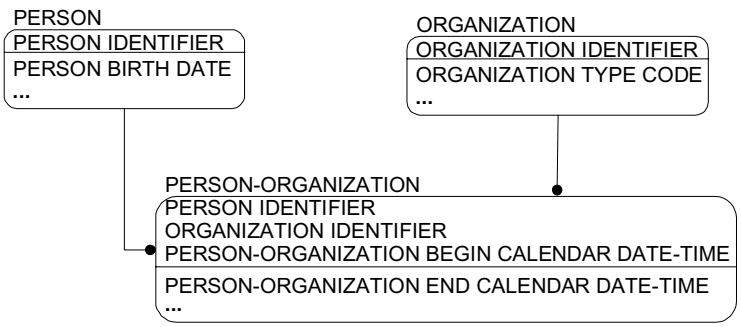


Figure 3. AICDM Representation of Many-to-Many Relationship

But organizations are not in the habit of describing their strength to their foes, so it is unrealistic to expect a data model to contain the names and weaponry of one's opponents. An organization can expect to receive intelligence on tidbits like approximate strength, at least as a raw number ("three armored battalions"), and a data model must capture it. The AICDM addresses this area by providing entities that model types rather than instances. Examples include:

- PERSON-TYPE, to model different categories of individuals (military vs. non-military; if military, the rank).
- ORGANIZATION-TYPE, to model different types of organizations (military vs. non-military; if military, the service, and distinguishing characteristics such as headquarters, vanguard, and rear guard).
- MATERIEL-ITEM, to model different types of materiel. The AICDM includes subtypes of MATERIEL-ITEM for common (and often complex) types of materiel, including WEAPON-TYPE and VEHICLE-TYPE).

The AICDM relates these entities to organizations, personnel, facilities, materiel, and features in one-to-many relationships that use an appropriate "holding" entity as an intermediate (See Figure 4). A holding entity describes the quantity of some entity type that an instance of an entity possesses. For example:

- The MATERIEL-ITEM-PERSON-HOLDING-ESTIMATE entity models an estimate of how many instances of a MATERIEL-ITEM a PERSON possesses. The attributes of the entity include not only the quantity but a statement of the estimate's accuracy.
- The PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE entity models an estimate of the number of people of a given type in a given organization.

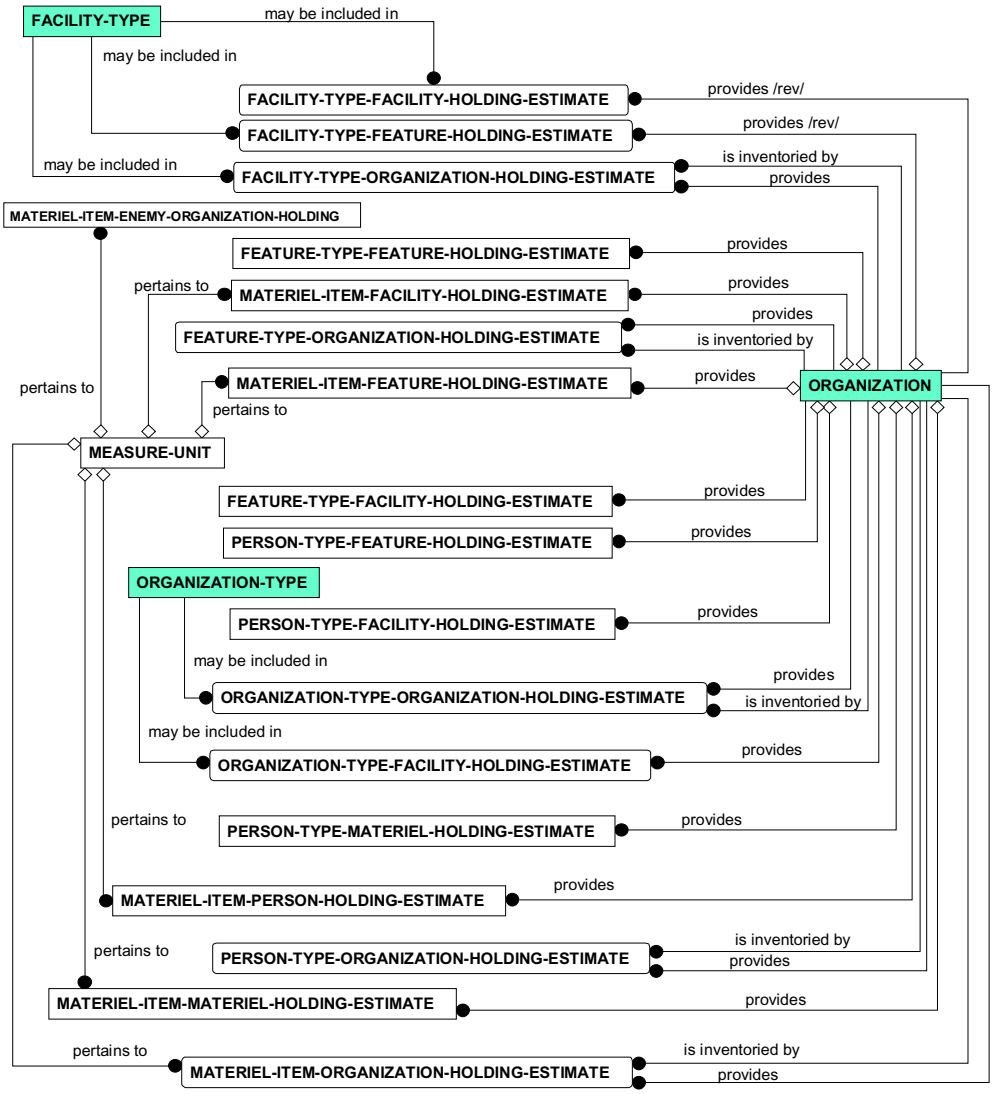


Figure 4. Examples of AICDM "HOLDING" Entities

3. Overview of the OMSC

The Object Management Standards Category (OMSC) was formed in April 1997 to develop standard objects for Army M&S functions. The OMSC is organized and maintained by the US Army Materiel Systems Analysis Activity (AMSAA). Its objective is to unify the many competing object models developed by Army and Joint offices. The intent is to provide "a high-level object class structure independent of any specific simulation environment." [AMSAA] The OMSC aims to give future M&S projects products that guide M&S architecture, design, and implementation through standards and software reuse.

3.1 OMSC Organization and Structure

The OMSC work products comprise a set of *standard objects*. Each standard object represents a recognizable concept commonly used in M&S systems. Standard objects extant at this time (in varying stages of completeness) include:

- Platform: A platform is any item that can be treated as an entity. Vehicles are examples of platforms. So are subcomponents of vehicles; that is, a platform is composed of platforms. An individual human is another example of a platform.
- Unit: A Unit is a military organization that represents a collection of entities. Units encompass organizations (companies, battalions, brigades, etc.) and functional groups (e.g., fire control centers).
- Location: A Location models information about the location of some entity or collection of entities.
- Behavior: A Behavior object provides a framework that encapsulates the definition of behavior within an M&S system.

Each standard object consists of a set of associated *classes*. In object-oriented software development, a class serves as a template for a set of object *instances*. Each instance models to an object, either a real-world physical object (e.g., a weapon), a real-world conceptual object (e.g., an organization), or a conceptual object defined during software development (e.g., a stack). OMSC classes model only real-world objects.

In standard objects defined so far, one of these classes bears the same name as the standard object, and can be considered as most intuitively representing the concept that the standard object models, capturing the central point of the standard object. It is the class that would be instantiated to represent an object instance. We term this class the *focal class*.

The focal class has associations with other classes in the standard object. Some of these classes in turn have associations with other classes. (The upshot is that the focal class has a transitive association to all classes in a standard object.) The OMSC uses two types of associations:

1. Aggregation: The OMSC uses this association to capture a situation where an instance of one class is associated with several instances of an other class. For example, a Unit may have several Communications capabilities.
2. Inheritance: The OMSC uses inheritance to capture subtyping relationships. For example, the OMSC defines Maintenance as a type of Logistics.

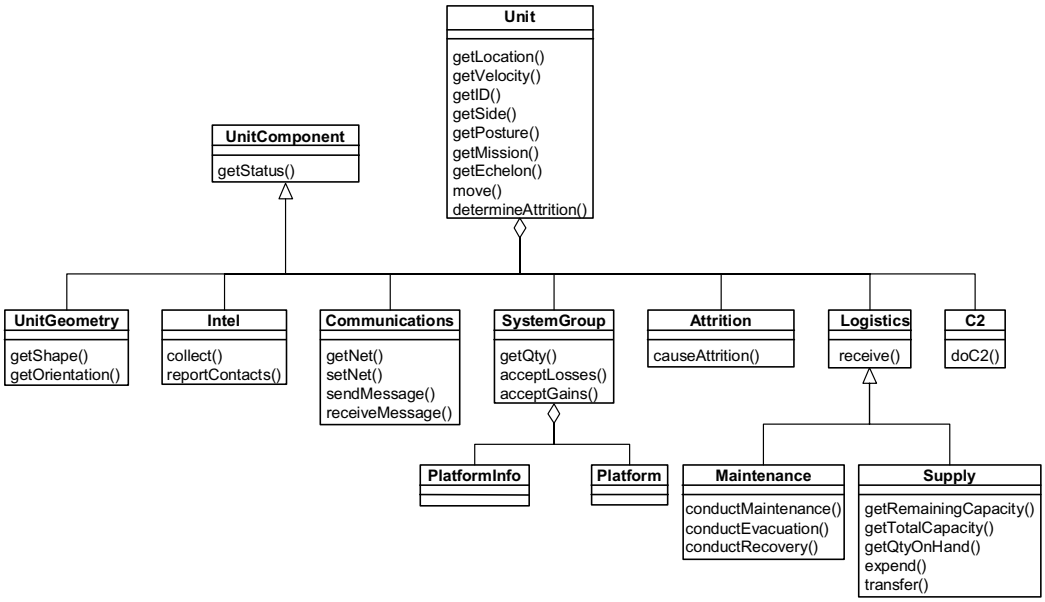


Figure 5. The OMSC Unit Standard Object

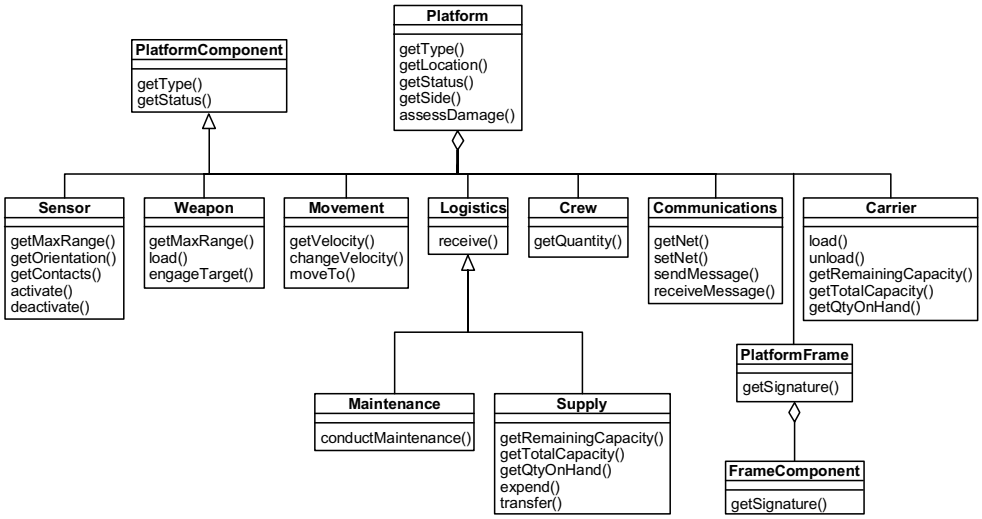


Figure 6. The OMSC Platform Standard Object

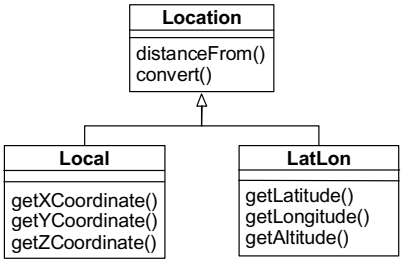


Figure 7. The OMSC Location Standard Object

The class diagrams for the completed OMSC standard objects—Unit, Platform, and Location—are presented in Figure 5, Figure 6, and Figure 7. These figures, which are taken from OMSC specifications, use the Unified Modeling Language (UML) [Booch 1996]. In UML, a class is represented by a rectangle. Associations between two classes are shown by connecting them with lines. If the line has a diamond, the class touching the diamond may have zero or more instances of the class at the other end of the line. If the line has a triangle, the class touching the triangle is a superclass of the class at the other end of the line.

The OMSC describes a class using three characteristics:

1. A class has a name. This name is suggestive of the role of the class.
2. A class has a prose description. ("Prose" means that automated analysis is impractical. Human analysis may suggest alignment strategies, but they can never be guaranteed. Prose can be aligned only to other prose.) It adds detail, and sometimes examples, to the name.
3. A class has a set of *methods*. In object-oriented programming, the methods of a class define the legal set of operations that may be performed on instances of a class. These operations govern exactly how instances may change over time, and what properties of an instance may be viewed at any time. The OMSC uses methods of both types. For example, the Unit object includes the following two methods:
    1. getLocation(), which yields the unit's current location.
    2. move(), which moves the object to a specified location.

The OMSC describes a method using a name and a (prose) description.

3.2 OMSC Status, Directions, and Issues

The OMSC standard objects are not, in their current state, complete models. Its creators have envisioned many standard objects but defined few. Initial plans called for six standard objects (Unit, Platform, Location, Environment, Data, and Behavior) by the end of 2000. Only Unit, Platform, and Location were completed. There are plans to add more standard objects to this set, but the objects have not been agreed upon and no schedule has been established. The Army recognizes that simulation encompasses much more than the OMSC currently describes.

The standard objects that have been defined are deliberately vague. The OMSC's charter is to create abstract objects that are simulation-independent, so its members do not want to constrain standard objects by tying them to a specific simulation environment. They chose to incorporate as many simulation systems as possible into their model. The classes in the OMSC provide high-level architectural guidance for a very broad range of M&S systems.

The OMSC provides little guidance below the architectural level, however. The many sources drawn on to define the OMSC standard objects have disparate implementations, and incorporating them into one model creates conflicts. Consider, for example, how to specify the operation of moving a unit. Does one state the final destination, or provide a direction of motion relative to the current location? Is the direction relative to the Earth's

geodetic coordinates, or to the unit's current orientation? And to what precision must a location be described? Different M&S systems answer each of these questions in their own way.

The OMSC's members decided to omit the detailed information that would make these conflicts surface. The information omitted includes the following:

- ***Method return values.*** There is no explicit and uniform indication of whether a method returns a value. Sometimes the method's description alludes to a returned value (for instance, getLocation() returns the current unit location). But the description usually does not state with any exactness the return value's semantics.
- ***Parameters for methods.*** The OMSC specifications never state the parameters required for methods. This leaves open many possible interpretations for state-modifying methods. For example, the description of the Unit's move() method says that it advances a unit toward its next location. It does not say how that location is determined, which is information that parameters would supply.
- ***Descriptions of data types.*** Many possible data types can be inferred from descriptions of classes or methods, but the OMSC does not give details on any of them. For example, the Geocentric class, which is a subclass of Location, has a method getLatitude(), which returns a latitude. The precision of this latitude is not stated.

But omitting details spells trouble for alignment efforts. We might perform a superficial analysis of movement in the two models and decide that they are aligned: after all, both the AICDM and the OMSC standard objects model location. But when we start examining details of movement, we find questions about the OMSC standard objects we cannot answer. Determining the degree to which two models are aligned is a matter of examining details. Details, unfortunately, are exactly what OMSC standard objects lack.

We address this problem by defining a model with four different levels of alignment. We use these levels to show what we might determine if the OMSC standard objects were more complete (covering all four levels), and then discuss what we can determine based on the OMSC's current state; the gap between "might" and "can" is, we shall see, considerable.

4. Definition of Alignment

This section describes in depth the meaning of alignment between the OMSC[2] and the AICDM. As will soon be explained, alignment can mean several things, depending on which elements one selects from the models being aligned. This section defines define four possible meanings, and chooses one as the focus of the analysis in this report.

The definition was created for studying alignment of the AICDM and the OMSC. It serves well as a definition of alignment between an ER model and an OO model that uses the modeling constructs of the OMSC (see Section 3.1). However, the OMSC makes limited use of OO modeling capabilities (see Appendix B). Without modification, the definition may not be optimal for studying alignment to other OO models.

Alignment is used to determine whether AICDM elements (views, entities, relationships, attributes, and domains) align with OMSC elements (standard objects, classes, and methods), and vice versa. What, then, does it mean to say elements are or are not aligned?

What alignment means depends upon the modeling elements used. The AICDM is an ER model. The OMSC standard objects use OO modeling. We can consider aligning entities in an ER model to classes in an object model; this is concerned (roughly) with matching things that model the same types of physical objects. Or we can consider aligning attribute domains in an ER model to data types in an object model. This is a set intersection issue.

The more formal the modeling elements, the more rigorous the definition of alignment. Set theory can be used to determine, rigorously, if an attribute domain and a data type are aligned. Their union must equal their intersection. But what does it mean to say an entity is aligned with a class? The informal definition above, about representing the same physical objects, still leaves room for interpretation.

We can think of a spectrum of alignment rigor. The top, highly abstract level comprises real-world concepts. At this level, we consider alignment intuitively. The bottom, highly detailed level has mathematical rigor suited to formal reasoning and computer-based implementations. In between are levels of increasing formality.

We identify four levels of alignment: Conceptual (level 1), Entity (level 2), State (level 3), and Value (level 4). They are summarized in Table 3. The rest of this section describes each level and defines what alignment between the AICDM and the OMSC means at that level. When we speak of alignment, we always refer to a particular level.

[2] In the remainder of this document ,“OMSC” is sometimes used as a shorthand for “OMSC standard object(s)”.

There is a pattern to the levels. In levels 1–3, level *i* contains a set of names of entities that are the focus of level *i*+1. In level 1, an OMSC standard object includes a set of class names; classes are the focus of level 2 in the OMSC. At level 2, an AICDM entity contains attribute names; attributes are the focus of level 3 in the AICDM.

Discussion of each level is broken into four parts:

1. Definition of level: An intuitive statement of the types of entities that make up the level.
2. Interpretation in the AICDM: The elements of the AICDM that appear in the level.
3. Interpretation in the OMSC: The elements of the OMSC that appear in the level.
4. Meaning of alignment: What it means to say that the AICDM and the OMSC are aligned with respect to the level.

Table 3. The Four Levels of Alignment

Level	Participating Model Entities	
	OMSC	**AICDM**
1 CONCEPTUAL: Entities of user perception	Standard Object • Name • Set of class names • Focal class name • Associations	View • Name • Set of entity names • Focal entity name • Relationships
2 ENTITY: Entities of data model	Class • Name • Description • Set of method names	Entity • Name • Description • Set of attribute names

3 STATE: Descriptions of entity state	Method
	• Name
	• Description
	• If method is behavioral:
	  • Set of parameters: name and data type (implicit in the OMSC)
	  • Optional return value data type (implicit in the OMSC)
	• If method is non-behavioral:
	  • Return value data type
		Attribute
		• Name
		• Description
		• Domain name

4 VALUE: Descriptions of domains	Data Type
	• Name
	• Set of values (discrete or continuous)
	• Relations:
	  • Comparison
	  • Order
		Attribute Domain
		• Name
		• Set of values (discrete or continuous)

So far alignment has been presented as black and white: entities are, or are not aligned. In fact the alignment definition recognizes shades of gray. Stating simply that the AICDM and the OMSC are not currently aligned is not useful. We want to know how closely they are aligned, and what steps to take to align them further. For this reason, the material on meaning of alignment covers degree of alignment. The degree of alignment is a rough measure of the number of entities that would need to be changed to achieve full alignment between the two models.

How one interprets degree of alignment depends, like alignment, on the level at which alignment is viewed. The intent here is that it provide a quantitative means to assess how a change to the AICDM or the OMSC affects alignment: closer, farther, or not at all. The definition given here yields a percentage. An AICDM entity and an OMSC entity might be assessed to be 25% in alignment.

Percentages do not suggest effort. They measure the number of entities in alignment, and say nothing about how alignment might be achieved. Going from 10% to 99% might require less effort than going from 99% to 100%.

For levels 1–3, degree of alignment is determined in terms of entities of the next lower level. Degree of alignment at the Conceptual level is determined based on entities in the Entity level; degree of alignment at the Entity level is determined based on entities in the State level; and degree of alignment at the State level is determined based on entities in the Value level. Degree of alignment at the Value level has a self-contained definition.

For levels 1–3, degree of alignment can also be defined independent of the next lower level. Such a definition is necessary if assessments at the next lower level are not possible (a situation that occurs more often than at the State level in the AICDM-OMSC alignment assessment). The sections on meaning of alignment discuss how to deal with these cases.

4.1 Conceptual Level

4.1.1 Definition of Level

A concept is a mental abstraction. The Conceptual level is concerned with the highest level abstractions one uses when thinking about a system, and the components of those abstractions. For instance, when someone thinks about an automobile, they think of wheels, a chassis, and an engine. When they think of a military unit, they think of a group of people and equipment. They may also think of properties of the system, such as the velocity of an automobile, and the location of a military unit.

The Conceptual level helps people imagine a system and its concept of operations. They want a general understanding of the entities[3] in the system. They are satisfied to assign a label such as “vehicle” or “unit” to each concept.

4.1.2 Interpretation in AICDM

The AICDM models the Conceptual level as a set of ER entity names, along with the names of the ER relationships that associate them, and an informal (textual) definition of intended semantics. In ER modeling, the usual name for this aggregate is a *view*. A view has a name that suggests what it models. Section 2.5 covered some of the current AICDM views.

One entity in a view is typically a “focal” entity, its name suggesting the whole concept. For example, the AICDM has a view named Organization. This view contains an entity

[3] The words “entity” and “relationship” are used here in their general sense, not as is meant in ER modeling. In this document, the meaning of the words will be clear from context, or explicitly stated.

named ORGANIZATION. It contains other entities with attributes relevant to modeling an organization, such as ORGANIZATION-OPERATIONAL-STATUS, COUNTRY, and MISSION. From the names of these entities, it should be clear that ORGANIZATION is the one most central to the view. ORGANIZATION, then, is the focal entity of the Organization view.

It is important to understand how the existing AICDM views are used in the alignment study. They serve as points of departure, not as abstractions against which the OMSC is compared. The existing AICDM views were not created to represent M&S data, so it would be unwise to expect that any given AICDM view should map exactly to some OMSC construct. During the alignment analysis, it was often useful to look at a predefined AICDM view instead of trying to tackle the entire AICDM model. However, the views defined for the purposes of Conceptual alignment are the creations of this study. They are drawn from the complete AICDM model, not from any single AICDM view, and they should not be taken to reflect the existing AICDM views.

4.1.3 Interpretation in the OMSC

The OMSC models the conceptual level as a standard object. A standard object has a name, and consists of a set of classes, plus associations among the classes. Each class has a name and an informal (textual) definition; nothing else about the class is relevant at the Conceptual level. Analogous to the AICDM, one class is typically a focal class. For example, the OMSC models a unit as a standard object of thirteen classes: Unit, UnitGeometry, Intel, Communications, etc. The associations relate Unit with the other classes in the standard object. See Figure 5.

The associations serve two purposes:

1. They express one to many relationships. One unit can have many platforms, for example.
2. They promote reuse of architectural concepts. Both units and platforms can have a communications capability. Rather than having both the Unit and Platform class contain identical communications methods, the OMSC defines a class Communication and associates it with both Unit and Platform. In this way, Communication is shared across standard objects.

4.1.4 Meaning of Alignment

The AICDM and the OMSC are fully aligned with respect to a concept when both can model that concept. An OMSC concept is modeled as a standard object, and an AICDM concept as a view. To say that the OMSC and the AICDM align with respect to a concept is to say that all of the following statements are true:

- There exists an OMSC standard object whose name has the same interpretation as that of an AICDM view. (We are free to create views, but the name of a view should reflect the elements it contains.)
- The focal class of the standard object is similar to the focal entity in the view.
- For each class in the standard object, there exists an ER entity (or set of entities) in the AICDM view that appears to model the same thing.
- For each aggregation association in the standard object, there exists a relationship in the view that captures the association (including cardinality). Since

many to many relationships in the AICDM are implemented using an intermediate entity, the aggregation association may map to two relationships and an entity.

Phrases in these statements—"suggests a similar purpose", "seems similar", "appears to model the same thing"—imply that Conceptual alignment is subjective. The wording is deliberate. With only the Conceptual level modeling elements (see Table 3), rigorous analysis isn't possible. In fact, rigorous analysis is only possible—and not necessarily guaranteed—at the Data Value level. All other levels have some degree of subjectivity. The Conceptual level is most subjective; the Data Value level least so.

As an example, the OMSC Platform standard object would align with an AICDM view consisting of entities such as MATERIEL that can model platforms. The Platform class is the focal class of the Platform standard object, and MATERIEL is the focal class of the view;[4] since they both model platforms, we consider the focal entities aligned. We also need to identify the AICDM entities that align with Weapon, Sensor, Communications, and other classes that are aggregated into a Platform, and we need to ensure that these entities have many-to-one relationships to MATERIEL (since one Platform can have zero or more Weapon, Sensor, Communications, etc. instances).

We can informally characterize the degree of alignment. A standard object implementing a concept consists of a set of classes. Each OMSC class, or each AICDM entity, has an associated percent degree of alignment, defined below. The degree of alignment for a standard object is the average of the degrees of alignment for each class in the standard object. (We define degree in terms of the OMSC because an OMSC standard object has a fixed number of classes. By contrast, in relational data models such as the AICDM each entity may ultimately be related to every other entity. The AICDM's notion of what constitutes a concept is softer than the OMSC's.) The alignment must occur between associations and relationships too. If the OMSC focal class has a one-to-many association with some other class, the AICDM focal entity must have a one-to-many relationship with the AICDM entity to which the class aligns.

We can also define Conceptual level degree of alignment independent of the Entity level, which is useful if we only want to analyze alignment at the Conceptual level. The degree of alignment of a standard object and a view is the percentage of classes that map to an AICDM entity.

4.2 Entity Level

4.2.1 Definition of Level

At this level, the focus is on the individual entities that make up a concept. Entity alignment is actually very similar to Conceptual alignment. Entity alignment is a necessary level in the alignment definition mainly in that it provides a means to define other levels of alignment.

[4] This example is somewhat simplified. See Appendix A.2 for a full discussion of the AICDM view that is aligned with the Platform standard object.

4.2.2 Interpretation in AICDM

The AICDM models an entity (in the general sense of the word) as a set of ER entities, with associated ER relationships. Usually the set will contain a single ER entity. However, ER design is not an exact science. Where one designer sees as a single ER entity as sufficient, another might argue for multiple ER entities associated by one to one relationships. We therefore allow multiple ER entities to model a single real-world entity.

The ER entity has a name, suggesting what it models, and a set of named attributes. The named attributes suggest characteristics of the ER entity.

4.2.3 Interpretation in the OMSC

The OMSC models an entity as a class. A class has a name and a set of methods, each also defined by name.

Some OMSC classes encapsulate not objects but a set of algorithms (Attrition is one example). These algorithms ultimately operate on objects, and it is these objects that are of concern to alignment.

4.2.4 Meaning of Alignment

The definition of alignment is almost the same as for the Conceptual level.

If the OMSC and the AICDM align with respect to some concept, then each OMSC class related to the concept is guaranteed to align to some set of AICDM entities and relationships related to the concept. The set of AICDM entities and relationships is some subset of that at the Conceptual level. For example, the AICDM MATERIEL entity aligns with the OMSC Platform class at the Entity level.

Some classes map to more than one ER entity, and vice versa. This is a consequence of the vagaries of modeling. If a class maps to more than one ER entity, the mapping must include relationships among the ER entities; and if an ER entity maps to more than one class, the mapping must include associations among the classes.

We define degree of alignment at the Entity level in terms of State level alignment below. Each State level alignment is defined by a percentage. Degree of entity level alignment is the average of the State level alignment percentages.

4.3 State Level

4.3.1 Definition of Level

The State level considers the behaviors and properties of entities. State alignment means one of two things:

- An entity in one model has the same properties (i.e., state) as an entity in the other model.

- If an OMSC operation affects an object instance, and there exist OMSC methods to determine that effect, then there exist AICDM attributes that can model the same information.

4.3.2 Interpretation in AICDM

The AICDM models state using:

- Organic attributes, i.e., attributes that are not migrated from another entity as a foreign key. PERSON HEIGHT DIMENSION is an organic attribute.
- Attributes from many-to-many relationships between entities that appear in associative entities. These attributes fall into three categories:
  - The foreign keys migrated from parent to child, which serve to record the existence of a relationship between two instances of entities. The ORGANIZATION-FACILITY entity, which captures the many-to-many relationship between ORGANIZATION and FACILITY, includes attributes FACILITY IDENTIFIER, ORGANIZATION IDENTIFIER, and ORGANIZATION-FACILITY IDENTIFIER.
  - Additional attributes needed to make an associative entity's key unique. The ORGANIZATION-FACILITY entity's key also includes ORGANIZATION-FACILITY IDENTIFIER.
  - Other attributes of an associative entity, which capture additional information about the relationship. The ORGANIZATION-FACILITY EFFECTIVE BEGIN CALENDAR DATE-TIME attribute is an example.

The distinction between the categories of attributes from many-to-many relationships is significant because the first and second categories exist as modeling artifacts to implement relationship existence, not to model application data. For instance, an M&S application might want to retrieve the ORGANIZATION-FACILITY EFFECTIVE BEGIN CALENDAR DATE-TIME attribute's value, but it should not need attributes of ORGANIZATION-FACILITY in the first two categories (such as ORGANIZATION-FACILITY-IDENTIFIER, which exists only to ensure that each record in the ORGANIZATION-FACILITY table is unique).

As just illustrated, attributes from many-to-many relationships often describe temporal properties. That a relationship exists between two entities implies a certain association during a specified period of time. Organic attributes, by contrast, are more often used to describe fundamental, unchanging properties of an individual entity.

4.3.3 Interpretation in the OMSC

In an object oriented model, state is queried or modified using methods. There are two types of methods:

1. A method with no side effects. Such a method is used to query the state of an object instance; it retrieves some property of the state. These methods tell us the type of state information associated with a class.
2. A method with side effects. Such a method is behavioral. The next invocation of one of the side-effect free methods will reflect the instance's altered state. These methods inform us about the types of operations an instance of the class can perform.

(A method may have side effects and also retrieve the altered state. We treat this as equivalent to having a side effect.)

At this level in the OMSC, a method is modeled by a name and a textual description of its purpose. We rely on intuition for a method's return type and parameters.

4.3.4 Meaning of Alignment

Alignment at the State level means that there is a suitable mapping between OMSC methods and AICDM attributes and relationships. More precisely:

- If an OMSC method has no side effects, then it aligns to a set of AICDM organic attributes and attributes from the relationships if the method's description implies the method returns a value that is modeled by the organic attributes and attributes from the relationships.
- If an OMSC method has a side effect, then it aligns to a set of AICDM organic attributes and attributes from the relationships if:
  - The method's description implies the method changes those organic attributes and attributes from the relationships.
  - If we can infer parameters for the method, the organic attributes and attributes from the relationships must include values for those parameters.

Sometimes an OMSC return value (or parameter) aligns to a single attribute, but often it aligns to more than one. This is a consequence of some combination of the following:

- A method returns (or uses) a composite value.
- A method operates on several classes of entities. For instance, the conductMaintenance() method of the Maintenance class operates on both platforms (equipment maintenance) and persons (medical treatment).

An OMSC method's return value or parameter can align to attributes and relationships either directly or indirectly. Direct alignment means there exists an AICDM attribute that stores the return (or parameter) value. For instance, the getLatitude() method of the OMSC's LatLon class returns a value that is stored by the POINT LATITUDE COORDINATE attribute.

Indirect alignment means the AICDM does not have an attribute that stores the method's return (or parameter) value, but the value can be computed based on other attributes and relationships. Consider the OMSC's Supply class, which has methods getTotalCapacity(), getQtyOnHand(), and getRemainingCapacity(). The getTotalCapacity() and getQtyOnHand() methods align directly, but the AICDM has no attributes that store remaining capacity. However, the result of getRemainingCapacity() can be computed as the difference between getTotalCapacity() and getQtyOnHand().

State Level alignment is defined with respect to Entity Level alignment. Side effect or no side effect, direct or indirect, method *M* of class *C* and attribute *A* of entity *E* can only be aligned at the State level if *C* and *E* are aligned at the Entity level (more exactly, if *E* is among the set of entities that are aligned with *C*).

Sometimes, a method's return value, or one of its parameters, aligns to a set of attributes that happen to be an AICDM entity. An example is the method getLocation(), which returns a value that aligns to the LOCATION entity. LOCATION is in turn the root of a subtype hierarchy; the hierarchy forms a view. In such situations, State level alignment is defined in terms of Conceptual alignment. In other words, getLocation() is aligned with LOCATION to the degree that LOCATION aligns with some AICDM view for locations.

Degree of alignment at the State level does not have a precise definition. That would require the OMSC to supply information on method parameters and return types. Instead, we define degree of alignment at the State level in terms of how clearly we can map what we expect are the method's parameters and return values to attributes of the ER entities aligned to the method's class. This mapping considers the following factors:

- Is there a single attribute, or are there several candidate attributes?
- How closely does the intent of the attribute appear to match the intent of the parameter/return value? (Sometimes we can define this in terms of Data Value alignment. For example, the OMSC Location class can model both geodetic and artificial two-dimension and three-dimensional spaces. The corresponding AICDM attribute in the Location entity can only describe geodetic references.)
- Is there a way to force-fit the mapping? Many AICDM attributes entities have attribute pairs in which one attribute stores a value and the other stores a code telling how to interpret that value. Sometimes, legal values for this code include "other" and "undefined". Either could be used to create—albeit poorly—a mapping.
- The degree of alignment for a method that aligns indirectly usually comes from the degree of alignment of other methods. For example, getRemainingCapacity() has a degree of alignment that is the minimum of the degrees of alignment of the two methods used to compute its value, getQtyOnHand() and getTotalCapacity().

4.4 Data Value Level

4.4.1 Definition of Level

The Data Value level considers the overlap of domains. The key issue is the degree to which values from a data type in one model can be mapped to another model. They may map partially, fully, or not at all.

4.4.2 Interpretation in AICDM

Each AICDM attribute has a data type. The data type is usually described according to standardized definitions contained in the Defense Data Dictionary System (DDDS).

4.4.3 Interpretation in the OMSC

The current version of the OMSC does not specify data type information.

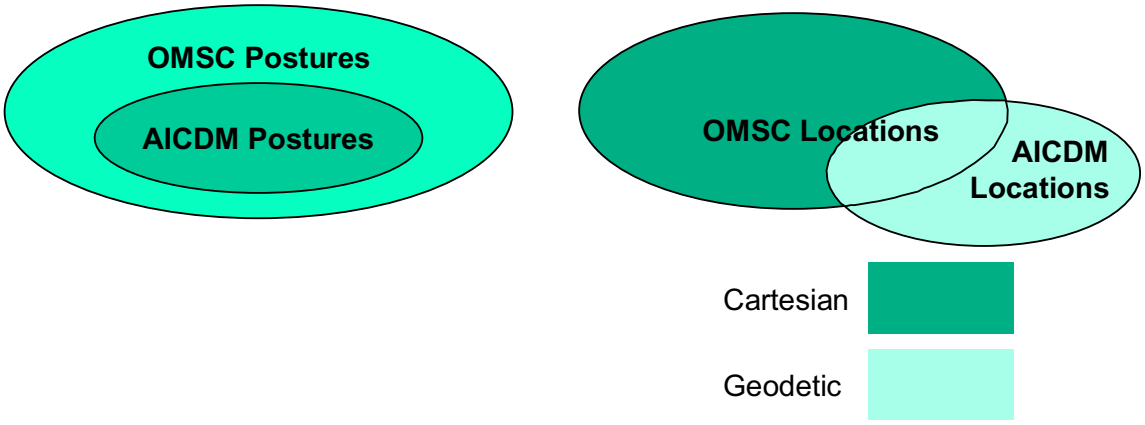
4.4.4 Meaning of Alignment

For a data value in one model to align fully with a value in the other model, both must have exactly the same data type and domain. It must be possible to represent exactly the same values, to the same degree of precision.

Also, an OMSC data value must be associated with a method that aligns with a corresponding AICDM attribute at the State level.

Degree of alignment is a function of the percentage of values that overlap. Consider an AICDM domain *A* and OMSC domain *O*. One might be a subset of the other. For example, the OMSC and the AICDM model postures, but the OMSC has a broader set of postures (see Figure 8-a). Or the two domains might have some values in common, but each may also possess unique values. For example, both the AICDM and the OMSC model locations, but only the OMSC models Cartesian coordinates, and the AICDM models geodetic coordinates to greater precision than the OMSC (see Figure 8-b).

Suppose both domains are discrete. Their degree of alignment is $|A \cap O| / |A \cup O|$, the cardinality of the intersection of the two domains divided by the cardinality of the union. Degree of alignment is a value between 0 and 1, with 1 being perfect alignment and 0 being no alignment.



a. Posture Domain Overlap b. Location Domain Overlap

Figure 8. Examples of Value Level Domain Overlap

The Value level is the deepest, most rigorous level. Only at this level can data alignment really be determined, in the sense of understanding whether fully automated translation between two data models is possible. But for the time being, working at the Value level is impossible. It requires detail that the OMSC does not begin to provide. We include the Value level to suggest future directions for determining alignment, and also as a guide to the completeness of the AICDM and the OMSC.

5. Assessing Alignment

This section covers the process we followed to determine alignment between the AICDM and the OMSC. The purpose of presenting the process is to allow for continued analysis of alignment as OMSC standard objects become available. Following the process helps ensure uniformity of analysis.

5.1 Overview

We performed analysis at the Conceptual, Entity, and State levels. We first determined alignment at the Conceptual level, used the results of Conceptual alignment as a guide in determining alignment at the Entity level, and then used the results of Entity alignment to guide alignment at the State level. We obtained, for each method, a numeric estimate of the degree of State level alignment between the AICDM and the OMSC with respect to that method. We then averaged these numeric estimates into an estimate of the degree of Entity level alignment between the AICDM and the OMSC with respect to each entity, and averaged the entity estimates into a value for the degree of Conceptual level alignment with respect to each OMSC standard object.

We started with OMSC standard objects and found corresponding AICDM views. We could have started with AICDM views and identified corresponding OMSC standard objects, but:

- Our task focused on identifying AICDM extensions which might be required to accommodate M&S data requirements as defined by OMSC standard objects. The OMSC therefore defines our task scope.
- The OMSC is much smaller than the AICDM. Most AICDM views don't map to OMSC standard objects, at least not yet. For the time being, starting with the OMSC narrows the task.
- The predefined AICDM views are broader than OMSC standard objects. Had we started with the AICDM, we would have found that any given view overlapped most of the OMSC. This was evident from quick examinations of each model even before we started the analysis.
- An object has a state, defined by its methods, and data/object model alignment determines the degree to which the data model captures states of the object model.

Note that alignment analysis is asymmetric. Using the OMSC AICDM orientation, we calculate average values based on (for example) the number of methods in a class. If we had chosen an AICDM OMSC orientation, we would have averaged the degrees of alignment of each attribute in an entity. There isn't a one-to-one correspondence between AICDM entities and OMSC classes, or between AICDM attributes and OMSC methods. Therefore, we obtain different sets of averages based on the orientation we choose. Furthermore, we deliberately ignore many AICDM attributes that aren't relevant to OMSC

methods. For example, the MATERIEL-ITEM entity, which records information about types of materiel, has a MATERIEL-ITEM PREFERRED INSPECTION PLACE CODE attribute that designates whether a type of materiel should be inspected at its source or destination. This attribute is irrelevant to simulations, at least as far as the OMSC is concerned, and would not align to any OMSC method. But alignment analysis looks for attributes that match methods, not methods that match attributes, so the absence of a method aligning to the MATERIEL-ITEM PREFERRED INSPECTION PLACE CODE attribute doesn't show up in the analysis.

5.2 Process

For each OMSC standard object, we identified the related AICDM view. In fact no predefined AICDM view exactly matches the intent of an OMSC standard object, so we noted the relevant AICDM entities and associations; these formed a view that served our purposes.

We mapped each OMSC class in a standard object to AICDM entities within the associated view. We mapped each OMSC method in a class to attributes within an entity. Sometimes the attributes were distributed across several entities. This occurred because of AICDM subtyping (e.g., subclasses of Location) or because the AICDM's designers opted to represent the concept described by an OMSC class as several entities. In either case, we noted the associations (inheritance, relationships, or both) among the entities.

At this point, the OMSC usually lacked detail and forced us to start making assumptions. We had to hypothesize the ranges of a method's return value and parameters.

Having made these assumptions, we were sometimes able to analyze the data values which we hypothesized an M&S system would use. The point of this analysis was not exactly to determine alignment. Instead, it was to investigate data values associated with the AICDM (determined through the DDDS). We often had to search the AICDM to determine if the an AICDM attribute mapped to what we thought would be a value returned or used by an OMSC method.

5.3 Example

The following example of alignment illustrates the process. It shows part of the analysis of alignment between the OMSC standard object Unit and the AICDM model. The results of the analysis are:

- A calculation of the degree of alignment between the standard object Unit and a view we define.
- Details on the alignment, down to the State level.

5.3.1 Conceptual Level

We begin by choosing conceptual level entities and analyzing how closely they are aligned. We choose the OMSC standard object Unit as the focus of this example.

The OMSC defines Unit as follows [HB 1998A]:

> A Unit encompasses military organizations that represent collections of entities (e.g., people, vehicles, weapon systems, etc.). Examples of this definition include organizations (i.e., companies, battalions, brigades, divisions, etc.) as well as functional groups (e.g., Tactical Operations Centers and Fire Control Centers).

We examine the Unit standard object (see Figure 5), looking for its focal class. The Unit class appears apt. It is clearly the root of a class hierarchy that includes aggregation and subtype associations. Its name, which is the same as that of the standard object, also suggests its central role.

We examine the AICDM model. We observe that it contains an entity named ORGANIZATION, which appears relevant. We further observe an entity named MILITARY-UNIT that is a subtype (though not directly) of ORGANIZATION. MILITARY-UNIT would appear to be an appropriate choice as the focal entity of the view we construct. See Figures A-8 and A-9.

We examine the predefined AICDM views and note that the Organization view contains both ORGANIZATION and MILITARY-UNIT. The Organization view appears to be a good starting point for building our view.

Looking through the OMSC classes in the Unit standard object, we see the need to model the concepts listed in Table 4.

Table 4. Concepts Suggested by the Unit Standard Object

Concept	Suggested By OMSC Class
Communications	Communications
Intelligence	Intel
Materiel	Logistics, Maintenance, and Supply
Geometry	UnitGeometry
People	Attrition
Platform	Platform and SystemGroup
C2	C2

The Organization view does not model all of these concepts. We examine the other predefined AICDM views and find that none fully captures the concept expressed by the Unit standard object.

We turn to the complete AICDM model, and add AICDM entities to the pre-existing Organization view to obtain the view we are defining, which we term the **Military Unit** view. Geometry is modeled by the LOCATION entity. The others are less certain. MATERIEL seems a reasonable match for Platform, although FACILITY might work too. The AICDM has no entity exactly matching intelligence; it does have INTELLIGENCE-RESOURCE EMPLOYEMENT. It also has the SENSOR-TYPE entity, which seems useful in intelligence collection.

The AICDM has many entities related to communications. All seem to be prefixed with TELECOMMUNICATIONS-NETWORK. Examples include TELECOMMUNICATIONS-NETWORK

ELEMENT and TELECOMMUNICATIONS-NETWORK DEVICE. We include these entities in our Military Unit view.

There are in fact other things in the AICDM that we need to align to the OMSC. For instance, we have assumed that attrition applies only to people, but in fact attrition can apply to both equipment and people. Nothing in the OMSC that we have studied so far has this much detail on attrition, so we don't know that yet. Entity level analysis reveals it.

5.3.2 Entity Level

A complete analysis of Entity level alignment involves an analysis of each OMSC class in the Unit standard object. In this example, we focus on the Unit class.

The OMSC Unit class aligns with the AICDM entity MILITARY-UNIT, the focal entity of our Military Unit view. Examining the names of Unit's methods, we can see the need for other AICDM entities and relationships if an instance of Unit is to be fully modeled in the AICDM, because the attributes of MILITARY-UNIT do not appear to encompass the range of concepts those methods comprise. So we add them, making certain they are related to a MILITARY-UNIT:

Table 5. Relationship of AICDM Entities to MILITARY-UNIT Entity

AICDM Entity	Suggested by OMSC Method	Relationship to MILITARY-UNIT
ORGANIZATION	N/A (supertype of focal entity)	MILITARY-UNIT is a subtype of UNIFORMED-SERVICE-ORGANIZATION. UNIFORMED-SERVICE-ORGANIZATION is a subtype of ORGANIZATION.[5]
LOCATION (and its subtypes)	getLocation() move()	MILITARY-UNIT is a subtype of ORGANIZATION. ORGANIZATION has a many-to-many relationship with FEATURE: • ORGANIZATION participates in ORGANIZATION-FEATURE. • FEATURE participates in ORGANIZATION-FEATURE. CONTROL-FEATURE is a subtype of FEATURE.[6] FEATURE has a many-to-many relationship with LOCATION: • FEATURE occupies FEATURE-LOCATION. • LOCATION locates FEATURE-LOCATION.

[5] For brevity, the remaining rows in this table omit the reference to UNIFORMED-SERVICE-ORGANIZATION.

[6] The CONTROL-FEATURE entity is used to record an organization's location. See Appendix A for details. An organization's location may also be described as follows: ORGANIZATION occupies ORGANIZATION-LOCATION; ORGANIZATION-LOCATION locates LOCATION. In the March 2001 version of the AICDM, either was acceptable. In more recent versions, a CONTROL-FEATURE is used to describe an organization's spread over an area, and an organization's center of mass is modeled as a a POINT. See Appendix E.

AICDM Entity	Suggested by OMSC Method	Relationship to MILITARY-UNIT
ORGANIZATION-OPERATIONAL-STATUS	getStatus() getPosture()	MILITARY-UNIT is a subtype of ORGANIZATION. ORGANIZATION is described by ORGANIZATION OPERATIONAL STATUS.
MISSION	getMission()	ORGANIZATION has a many-to-many relationship with MISSION: • ORGANIZATION is referenced in MISSION-ORGANIZATION. • MISSION is controlled by MISSION-ORGANIZATION.
ORGANIZATION-TYPE-ORGANIZATION HOLDING ESTIMATE	determineAttrition()	MILITARY-UNIT is a subtype of ORGANIZATION. ORGANIZATION provides ORGANIZATION-TYPE-ORGANIZATION-HOLDING-ESTIMATE.
PERSON-TYPE-ORGANIZATION HOLDING ESTIMATE	determineAttrition()	MILITARY-UNIT is a subtype of ORGANIZATION. ORGANIZATION provides PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE.

We note an ambiguity at this stage of the analysis. The OMSC states that a location is typically a point representing a center of mass. Apparently it can be something other than a point. Perhaps it is one of the geometric figures modeled by the AICDM entity REGULAR-AREA. For this reason we include all subtypes of LOCATION.

5.3.3 State Level

We continue the alignment analysis at the State level by examining each method of the Unit class.

The getLocation() Method

From the method's description, we infer that it returns a value that represents a Location. The OMSC defines a LOCATION class hierarchy. This hierarchy constitutes another standard object. We therefore revert to Conceptual level alignment analysis to determine if there exists, or if we can define, an AICDM view that aligns with Location. We discover that the OMSC Location standard object partially aligns with a view consisting of the AICDM's LOCATION entity and its subtypes.

The getVelocity() Method

This method's description indicates that it returns the velocity of a Unit. We fail to find any mapping to the AICDM, which does not model motion of organizations. We conclude there is no alignment at the State level between getVelocity() and the AICDM.

The getID() Method

We find some ambiguity in aligning the getID() method, because the AICDM has many attributes that might be relevant. We cannot ascertain which (if any) is appropriate without more detail from the OMSC. This cannot be resolved except at the Value level, and the OMSC does not supply the necessary information to conduct analysis at that level.

The getID() method might map to any of the following AICDM attributes of MILITARY-UNIT:

- MILITARY-UNIT ALTERNATE IDENTIFIER
- MILITARY-UNIT REFERENCE NUMBER ID
- MILITARY-UNIT TROOP PROGRAM IDENTIFIER
- MILITARY-UNIT FOREIGN MILITARY TRANSLATED NAME
- MILITARY-UNIT IDENTIFIER

The getSide() Method

Examining the attributes of AICDM entities, we find the AICDM does not model sides in the same way as the OMSC. The AICDM has attributes that allow an organization to be designated as friend, foe, neutral, or various other values. It also has an entity ENEMY-ORGANIZATION. The problem is that it does not allow for arbitrary sides. The AICDM does not explicitly model coalitions and their allegiances.

However, getSide() does not specify enmity. In other words, the AICDM does not need to model allegiances to align with getSide(). It only needs to model the existence of sides. The AICDM can do this indirectly through the ORGANIZATION-ASSOCATION entity, which has a relationship with ORGANIZATION. The relationship is used to model organizational hierarchies. Since a coalition (or faction) is a type of hierarchy, we can model a coalition (or faction) as an ORGANIZATION, related to other ORGANIZATION entities via an ORGANIZATION-ASSOCIATION entity:

- We create an ORGANIZATION as the root of the hierarchy. The ORGANIZATION-IDENTIFIER attribute contains the name of the faction or coalition.
- For each ORGANIZATION that is a member of the faction or coalition, we relate it to the root ORGANIZATION via an association through an instance of the ORGANIZATION-ASSOCIATION entity:
  - ORGANIZATION is the subject of ORGANIZATION-ASSOCIATION
  - ORGANIZATION is the object of ORGANIZATION-ASSOCIATION

Note that nothing explicitly identifies the root ORGANIZATION as denoting a coalition. The DESCRIPTION TEXT attribute could be used, but that's just prose. None of the other attributes have a suitable value (though we don't discover this until we analyze alignment at the Data level).

The getPosture() Method

The OMSC is rather vague about the definition of posture. But then, so is the Army. In any case, the AICDM entities have several attributes that seem to offer the possibility to model posture:

- ORGANIZATION-OPERATIONAL-STATUS PROTECTIVE POSTURE CODE
- ORGANIZATION-OPERATIONAL-STATUS CURRENT ACTIVITY CODE
- MILITARY-UNIT CURRENT ACTIVITY CODE

However, none of these names seems to suggest the ability to accommodate the full range of postures suggested by the descriptive text for the method. We note the possibilities and wait until we perform Data level analysis.

The getStatus() Method

The OMSC is also vague about status. In the AICDM the following attributes are possibilities, but none of the names seem to suggest the ability to accommodate the full range of statuses suggested by the descriptive text for the method:

- ORGANIZATION-OPERATIONAL-STATUS COMBAT EFFECTIVENESS CODE
- ORGANIZATION-OPERATIONAL-STATUS COMBAT READINESS CODE
- MILITARY-UNIT CURRENT READINESS CONDITION CODE
- MILITARY-UNIT DEFENSE READINESS CONDITION CODE

The getMission() Method

We read the description of this method and realize that, in M&S systems, a mission is prose. This contradicts our earlier assumption that the result of getMission() aligns with a MISSION (see Table 5). A MISSION comprises a set of TASK entities. The TASK entities have a free text attribute (TASK-DESCRIPTION-TEXT). The getMission() method aligns better with TASK than with MISSION.

The alignment is unidirectional. An AICDM TASK may be a hierarchy. We might map this to an AICDM mission using some standard algorithm for amalgamating the task descriptions in the hierarchy, but we couldn't necessarily recover the original structure from getMission().

The getEchelon() Method

The return value of this method maps to the MILITARY-ORGANIZATION UNIT-LEVEL CODE attribute.

The move() Method

This is a behavioral method. The OMSC does not specify any parameters, but movement is directed and so the method must obtain its destination from somewhere. It seems safe to assume that most M&S systems will invoke the method with parameters.

There are two obvious parameter schemes that could be used. The parameters could consist of:

1. A new location, probably specified as a Location object (i.e., as a Cartesian coordinate). Possibly it could be stated as a name (e.g., "Rockefeller Center") but that would mean the Unit class must interface with a mapping system. The OMSC descriptions of Unit haven't provided any evidence thereof.
   The AICDM can partly model a Location object, as noted above in the description of getLocation(). A LOCATION, then, would sometimes be a suitable mapping of the parameter to move() under this scheme.
2. A vector, relative to the current location. A unit might be instructed to move 100 yards forward, for instance.
   Whether the AICDM can model this scheme depends on how the direction is specified. The AICDM does not model organization alignment and so cannot model directions such as "forward" that are relative to alignment.

There is also no entity in the AICDM that captures the concept of relative motion. Perhaps an OMSC specification of relative motion would map to an AICDM TASK. But a TASK is more general than just a motion specification, and so cannot be mapped back.

In either case, the AICDM must be able to model the state before and after the method is invoked.

The change in state consists of a change to a unit's location. The AICDM is able to model this change, because an ORGANIZATION has an associated LOCATION. The association is through relationships involving an entity ORGANIZATION-LOCATION, which has attributes ORGANIZATION-LOCATION EFFECTIVE CALENDAR DATE and ORGANIZATION-LOCATION EFFECTIVE TIME. These attributes are particularly useful in keeping a history of state, if desired.

The determineAttrition() Method

This is a behavioral method. The description states that it calculates attrition caused by another unit or platform. We can infer the following:

1. Attrition is calculated with respect to some previous state of the unit. The duration between the time of that state and the present is either fixed or specified as a parameter.
   In either case, the time of the previous state would align to an AICDM attribute. The associations between an ORGANIZATION and its personnel or materiel include attributes that record the date and time during which the association applies.
2. The method's description refers to "another unit or platform" in the singular. This implies that attrition is calculated with respect to a single unit or platform. Since simulations have many units and platforms, the one that causes attrition would need to be a parameter of determineAttrition().
   We therefore expect that at least one parameter of determineAttrition() would align to a MILITARY-UNIT or MATERIAL-ITEM of the AICDM.

The change in state caused by determineAttrition() will be the loss of personnel, materiel, etc. from a unit. How this is done depends on whether the simulation models individual entities or quantity of entities; e.g., whether it records that a unit has tanks identified A, B, and C or simply notes that the unit has 3 tanks. If it models individual entities, then the change in state aligns to the following AICDM attributes:

- PERSON-ORGANIZATION BEGIN CALENDAR DATE-TIME
- PERSON-ORGANIZATION END CALENDAR DATE-TIME
- MATERIEL-ORGANIZATION (N.B. MATERIEL-ORGANIZATION is an entity, and it lacks a date-time attribute to record when the association exists.)

If the simulation models entity quantity, the change in state aligns to the following AICDM attributes:

- PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE QUANTITY,
  PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE CALENDAR DATE-TIME
- MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE QUANTITY,
  MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE CALENDAR DATE-TIME

The AICDM can model more types of quantity changes than associations with individual entities. However, modeling attrition of facilities, features, and organizations may not be important.

The effect of determineAttrition() might also be modeled using the AICDM ACTION-EFFECT entity. However, a decision was made to postpone doing so until the OMSC has a better-defined model of behavior. See p.31.

5.3.4 Value Level

We cannot, as a rule, determine alignment at this level. The OMSC lacks detail. We can, however, make inferences based on statements in the OMSC and the known values for the AICDM. Mostly this is a matter of understanding whether the range of possible values for the AICDM captures the intent—either stated by the OMSC or inferred by us—of OMSC values. Table 6 gives some examples of inferences. The first column lists values (i.e., data types) returned by OMSC methods. The second column suggests AICDM entities and attributes with which these values might align. The third column gives issues: inferences and ambiguities.

Table 6. Data Level Alignment Issues

Value	Alignment Possibilities	Issues
side	Unit.getSide() to ORGANIZATION	If we adopt the convention for modeling sides used on p. 38, there is no value for ORGANIZATION TYPE CODE that explicitly identifies an organization as denoting a coalition or faction. We would need to use either NOT SPECIFIED or NOT KNOWN. Also, there is no clearly suitable value for ORGANIZATION-ASSOCIATION TYPE CODE in expressing a coalition or faction. Values such as IS MOBILIZED TO or IS ATTACHED TO might work in some circumstances.
posture	Unit.getPosture() to ORGANIZATION-OPERATIONAL-STATUS PROTECTIVE POSTURE CODE	The AICDM postures seem restrictive in comparison to the OMSC postures
status	Unit.getStatus() to ORGANIZATION-OPERATIONAL-STATUS COMBAT EFFECTIVENESS CODE	The AICDM status seems restrictive in comparison to the OMSC status

5.3.5 Degree of Alignment

We shall determine degree of alignment with respect to the top 3 levels. The process is to work bottom-up. Starting at the State level, we assign to each method a degree of alignment. As noted in Section 4.3.4, we assign a value informally, because we lack Value level analysis data.

The assigned value is a percentage. We opt to use five values (more values are difficult to justify). These values are explained in Table 7. (The “Standard Phrase” column is particu-

larly important in the alignment analysis in Appendix A. It relates the analysis, which is free text, to the numeric value.)

Table 7. Possible Degrees of Alignment

Value	Standard Phrase	Meaning
0%	No alignment	This value is assigned in either of the following circumstances: • There is no overlap between the models. One model contains an instance of an element that has no analog in the other. • Lack of information in the OMSC prevents alignment analysis.
25%	Low degree of alignment	There is some overlap, but it seems coincidental. Overlap might have been achieved by using AICDM attributes in ways that its designers did not originally intend.
50%	Medium degree of alignment	There is a moderate amount of overlap, but still a significant disconnect between the models.
75%	High degree of alignment	Perfect alignment can probably be achieved by small changes to one model or the other.
100%	Perfect alignment	There is an exact, unambiguous mapping between the models.

Table 8 shows, for each method, its degree of alignment to the AICDM. The table gives a rationale for each number. This rationale is a summary of the information in Appendix A, which fully describes why the value is merited. (Note that the degree of alignment in the first row is not assigned, but calculated, which is why it's not one of the values from the first column of Table 7.)

Table 8. State Level Degree of Alignment

Method	% Alignment	Rationale
getLocation()	37%	The value getLocation() returns is represented by a standard object whose degree of alignment is calculated elsewhere to be 37%. See Section 6.3.
getVelocity()	0%	The AICDM cannot model velocity.
getID()	75%	The AICDM has several candidate attributes for ID. A unique mapping cannot be determined.
getSide()	75%	Sides may be modeled in the AICDM through organization associations. However, if there are more than two sides, there is no clear way to express all possible combinations of friend/foe/neutral relationships.
getPosture()	25%	The range of values given for the AICDM attributes that represent posture seems small compared to the examples given for the OMSC.
getStatus()	25%	The range of values given for the AICDM attributes that represent posture seems very small compared to the examples given for the OMSC.
getMission()	50%	An OMSC mission can be mapped onto an AICDM TASK. However, an AICDM MISSION can be a complex structure that cannot necessarily be mapped onto an OMSC mission. At least, the mapping is not reversible.

Method	% Alignment	Rationale
getEchelon()	100%	The OMSC and the AICDM both model echelons. The AICDM's values are a little broader, but a given M&S system can define an exact mapping.
move()	50%	Some forms of move() can be aligned, but some can't, depending on the parameter configurations.
determineAttrition()	75%	determineAttrition() aligns indirectly. It aligns to the same degree as the Attrition.causeAttrition() method.

We calculate degree of alignment at the Entity level based on this information. The degree of alignment between the OMSC Unit class and the AICDM entities and relationships identified above is the average of the degrees of alignment of the methods associated with the Unit class:

$$\frac{37+0+75+75+25+25+50+100+50+75}{10} = 51\%$$

The degree of alignment between the Unit standard object and the AICDM Military Unit view is calculated as the average degree of alignment between classes in the Unit standard object and sets of AICDM entities and relationships, in a similar fashion.

Table 7 uses a scale between 0 and 100 and provides for the use of 5 values on that scale. To cover the scale (which is necessary if calculated values like 51% are to be introduced), it is necessary to interpret a value from Table 7 as covering a range of possible degrees of alignment, as shown in Table 9.

Table 9. Statistical Interpretation of Degrees of Alignment

Value	Mean	Range
0	6.25	$0 < X \leq 12.5$
25	25	$12.5 < X \leq 37.5$
50	50	$37.5 < X \leq 62.5$
75	75	$62.5 < X \leq 87.5$
100	92.75	$87.5 < X \leq 100$

Table 9 shows the values 25, 50, and 75 as mean ranges that cover ranges ±12.5. The values 0 and 100 have means of 6.25 and 92.75 respectively, and cover ranges ±6.25, since values less than 0 or greater than 100 are not possible.

The meaning of the ranges is that a method's degree of alignment has some error. This error is due to the coarseness of the scale in Table 7. When an analyst assigns a degree of alignment of 50% to a method, he is really saying that the degree of alignment is between 37.5% and 62.5% The existence of this error implies that the value of 51% computed for the Unit class has some error. However, we can estimate that error by making the following assumptions:

- Errors are uncorrelated.
- The errors in selecting a value from within a range are normally distributed.

These assumptions are not strictly true (see Appendix A), but the sample sizes used in the calculations in Section 6 and Appendix A are large enough that the effect shouldn't matter.

The central limit theorem of statistics says that whenever a random sample of size *n* with mean μ and variance σ^2 is taken from a distribution, the sample mean $\overline{X}_n$ has a distribution that is approximately normal with mean μ and variance σ^2/n. In other words, the variance (and hence the standard deviation) is inversely proportional to the sample size.

The second assumption means the standard deviation for a range *r* is $r/\sqrt{3}$. Therefore, the standard deviation is $6.5/\sqrt{3}$ for alignment values 0 and 100, and $12.5/\sqrt{3}$ for 25, 50, and 75. The standard deviation for the Location standard object is 3.61 (see Section 6.3), so the standard deviation for the error in the degree of alignment of the Unit class is:

$$\frac{3.61+2\times\left(6.25/\sqrt{3}\right)+7\times\left(12.5/\sqrt{3}\right)}{10}\approx 6$$

In other words, the probability that the degree of alignment of the Unit class is between 45% and 57% (within 1 standard deviation of the mean) is about 68%.

This analysis describes the error a single analyst can make in calculating degree of alignment. Since the assignment of values at the State level has inherent subjectivity, two analysts might assign the same method different degrees of alignment. The assumptions made to compute error cover estimating error only.

6. Results of Applying the Model

This section presents the results of assessing alignment between the AICDM and the OMSC using the process described in Section 5: a calculation of the degree of alignment between the AICDM and the OMSC for each standard object in the OMSC. Briefly, these results are as follows:

Table 10. Standard Object Degrees of Alignment

Standard Object	Degree of Alignment to the AICDM
Unit	56%
Platform	55%
Location	37%

Degree of alignment is calculated with respect to the Conceptual, Entity, and State levels:

For each standard object:
For each class in the standard object:
For each method in the class:
- If the analysis for the method in Appendix A indicates the method aligns to attributes (directly or indirectly), then estimate the method's degree of alignment to those attributes.
- If the analysis indicates the method aligns to an AICDM entity or view, use the degree of alignment calculated for that entity or view.

Compute the class' degree of alignment as the average of the degrees of alignment of all methods in the class.
Compute the standard object's degree of alignment as the average of the degrees of alignment of all classes in the standard object.

Each standard object has a separate section. Each class within a standard object has a subsection. Within that subsection is a table giving degrees of alignment for all the class' methods. The table explains the rationale for the estimated degree of alignment. The rationale is a summary of the materiel on the method in Appendix A.

The OMSC shares classes among standard objects (e.g., Communications). For a shared class, the degree of alignment calculation is presented once, then referenced for other uses of the class.

The error can be computed for each class (see p. 43). But some classes contain only one or two methods, and error computations would be suspect. It makes more sense to compute the error for an entire standard object. These values are given at the end of each standard object's subsection.

6.1 Unit Standard Object

6.1.1 Unit Class

Method	%	Rationale
getLocation()	*37%*[7]	(See Section 0 for the derivation of this degree of alignment)
getVelocity()	0%	The AICDM cannot model velocity.
getID()	75%	The AICDM has several candidate attributes for ID. A unique mapping cannot be determined, however.
getSide()	75%	Sides may be modeled in the AICDM through organization associations. However, if there are more than two sides, there is no clear way to express all possible combinations of friend/foe/neutral relationships.
getPosture()	25%	The range of values given for the AICDM attributes that represent posture seems small compared to the examples given for the OMSC.
getStatus()	25%	The range of values given for the AICDM attributes that represent posture seems very small compared to the examples given for the OMSC.
getMission()	50%	An OMSC mission can be mapped onto an AICDM TASK. However, an AICDM MISSION can be a complex structure that cannot necessarily be mapped onto an OMSC mission. At least, the mapping is not reversible.
getEchelon()	100%	The OMSC and the AICDM both model echelons. The AICDM's values are a little broader, but a given M&S system can define an exact mapping.
move()	50%	Some forms of move() can be aligned, but some can't, depending on the parameter configurations.
determineAttrition()	75%	determineAttrition() aligns indirectly. It aligns to the same degree as the Attrition.causeAttrition() method.

51%	**Unit class Degree of Alignment**

6.1.2 UnitGeometry Class

Method	%	Rationale
getShape()	100%	The AICDM can model arbitrary polygons, and seems to have an adequate model of three-dimensional entities.
getOrientation()	0%	The AICDM does not model orientation.

50%	**UnitGeometry class Degree of Alignment**

[7] This and some other numbers are in *italics* to indicate that they are not drawn from Table 7, but calculated as the degree of alignment of some other entity.

6.1.3 Intel Class

Method	%	Rationale
collect()	0%	Nothing about how the collect() method aligns to the AICDM can be determined at the State level.
reportContacts()	75%	A contact is probably a battlefield entity (FEATURE, PERSON, MATERIEL, ORGANIZATION, or FACILITY), but the OMSC does not provide details.

37%	**Intel class Degree of Alignment**

6.1.4 Communications Class

Method	%	Rationale
getNet()	100%	The AICDM can model objects capable of exchanging messages, which is all that must be returned or set by these methods.
setNet()	75%	The AICDM can model more types of communications networks than the OMSC.
sendMessage()	75%	The content of a message can be modeled as an INFORMATION-REFERENCE, and the act can be modeled through the AICDM ACTION entities. However, the AICDM lacks attribute values to model sending or receiving messages.
receiveMessage()	75%	

81%	**Communications class Degree of Alignment**

6.1.5 SystemGroup Class

Method	%	Rationale
getQty()	100%	These methods' return values have exact counterparts in AICDM attributes.
acceptLosses()	100%	
acceptGains()	100%	

100%	**SystemGroup class Degree of Alignment**

6.1.6 Platform Class

This class is modeled by an OMSC standard object. See Section 6.2.

55%	**Platform class Degree of Alignment**

6.1.7 PlatformInfo Class

The description of the PlatformInfo class indicates that it returns static properties of entities. The AICDM models very few static properties of entities.

25%	**PlatformInfo class Degree of Alignment**

6.1.8 Attrition Class

Method	%	Rationale
causeAttrition()	75%	The AICDM's model accounts for both changes in associations among entities and for holdings of materiel and personnel types. However, if the attrition is expressed as holdings, the AICDM can only model loss. Its model of degradation is poor.

75%	**Attrition class Degree of Alignment**

6.1.9 Logistics Class

Method	%	Rationale
receive()	75%	The attributes used to record logistics information are a subset of those used to determine attrition. Since the Attrition class aligns 75% with the AICDM, the receive() method must align 75% too.

75%	**Logistics class Degree of Alignment**

6.1.10 Maintenance Class

Method	%	Rationale
conductMaintenance()	25%	The AICDM attributes that record operational status of a person or materiel item do not fully capture the meaning of what is expressed by this operation.
conductEvacuation()	50%	The MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE and PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE entities represent the number of materiel items and personnel associated with an ORGANIZATION. However, these entities have no relationships to the LOCATION entity, which models the location of an entity. This would be a problem if only part of an organization were evacuated. (One strategy for modeling evacuation would be to create a new ORGANIZATION for each group of evacuated personnel and equipment.) The MATERIEL entity has a many-to-many relationship with LOCATION and with ORGANIZATION. However, MATERIEL models individual materiel items, not types.
conductRecovery()	50%	

42%	**Maintenance class Degree of Alignment**

6.1.11 Supply Class

Method	%	Rationale
getRemainingCapacity()	75%	This method aligns indirectly. Its degree of alignment is the minimum of those of getTotalCapactity() and getQtyOnHand().
getTotalCapacity()	75%	The AICDM can model total capacity by relating a CAPABILITY entity to an ORGANIZATION via ORGANIZATION-CAPABILITY-ESTIMATE or ORGANIZATION-TYPE-CAPABILITY-NORM. However, the values of the CAPABILITY TYPE CODE attribute are inadequate.
getQtyOnHand()	75%	This method aligns only if its result is an ordinal value. If it is measured in other units, the AICDM has no attributes that align.
expend()	75%	These methods align to the same degree that getQtyOnHand() does.
transfer()	75%	

75%	**Supply class Degree of Alignment**

6.1.12 C2 Class

Method	%	Rationale
doC2()	0%	Nothing about alignment can be determined.

0%	**C2 class Degree of Alignment**

6.1.13 Unit Standard Object Degree of Alignment

The degree of alignment for the Unit standard object is the average of the degrees of alignment of the classes it contains:

$$\frac{51+50+37+81+100+55+25+75+75+42+75+0}{12} = 56\%$$

The error in the degree of alignment is:

$$\frac{4+6+10\times\left(6.25/\sqrt{3}\right)+22\times\left(12.5/\sqrt{3}\right)}{34} \approx 6$$

(4 and 6 represent the errors from the Location and Platform standard objects, respectively. The Unit standard object contains 10 methods whose degrees of alignment are assigned either 0 or 100 and therefore have a standard deviation of $6.25/\sqrt{3}$. The other 22 methods have a standard deviation of $12.5/\sqrt{3}$.)

6.2 Platform Standard Object

6.2.1 Platform Class

Method	%	Rationale
getType()	75%	The OMSC is vague about type. The analysis has indicated that some types can be modeled, but it is unrealistic to expect that all can.
getStatus()	100%	The OMSC examples of status for platforms are considerably simpler than for units. In this case, the AICDM attributes appear adequate.
getLocation()	*37%*	See the description of getLocation() in Section 6.1.1.
getSide()	75%	See the description of getSide() in Section 6.1.1.
assessDamage()	25%	The AICDM does not have attributes designed to assess damage. The analysis found a few attributes whose domains can designate certain broad classes of damage.

62%	**Platform class Degree of Alignment**

6.2.2 Sensor Class

Method	%	Rationale
getMaxRange()	100%	This method aligns exactly to an AICDM attribute.
getOrientation()	0%	The AICDM does not model orientation.
getContacts()	50%	The vagueness of the OMSC description of the method leaves open the possibility that this method returns information the AICDM cannot model.
activate()	0%	AICDM attribute values apparently model capability to operate rather than operation.
deactivate()	0%	

30%	**Sensor class Degree of Alignment**

6.2.3 Weapon Class

Method	%	Rationale
getMaxRange()	100%	This method aligns exactly to an AICDM attribute.
load()	0%	The AICDM does not model whether a weapon is loaded.
engageTarget()	50%	The AICDM does not model properties of a weapon related to target engagement.

50%	**Weapon class Degree of Alignment**

6.2.4 Movement Class

Method	%	Rationale
getVelocity()	0%	The AICDM does not model velocity.
changeVelocity()	0%	
moveTo()	75%	At least some forms of movement can be modeled, probably more than for a unit (see the description of move() in Section 6.1.1).

25%	**Movement class Degree of Alignment**

6.2.5 Logistics Class

The degree of alignment of the Logistics class is equal to the degree of alignment of the Logistics class in the Unit standard object. See Section 0.

75%	**Logistics class Degree of Alignment**

6.2.6 Supply Class

The degree of alignment of the Supply class is equal to the degree of alignment of the Supply class in the Unit standard object.

75%	**Supply class Degree of Alignment**

6.2.7 Maintenance Class

Method	%	Rationale
conductMaintenance()	25%	The AICDM attributes that record operational status of a person or materiel item do not fully capture the meaning of what is expressed by this operation.

25%	**Maintenance class Degree of Alignment**

6.2.8 Crew Class

Method	%	Rationale
getQuantity()	100%	This method is modeled either by direct relationships or by HOLDING entities.

100%	**Crew class Degree of Alignment**

6.2.9 Communications Class

The degree of alignment of the Communications class is equal to the degree of alignment of the Communications class in the Unit standard object. See Section 0.

81%	**Communications class Degree of Alignment**

6.2.10 Carrier Class

Method	%	Rationale
load()	100%	These methods are modeled either by direct relationships or by HOLDING entities.
unload()	100%	
getRemainingCapacity()	75%	This method aligns indirectly. Its degree of alignment is the minimum of that of getTotalCapacity() and getQtyOnHand().
getTotalCapacity()	75%	See Section 0.
getQtyOnHand()	75%	See Section 0.

85%	**Carrier class Degree of Alignment**

6.2.11 PlatformFrame Class

Method	%	Rationale
getSignature()	25%	The AICDM can model a very limited range of signature types.

25%	**PlatformFrame class Degree of Alignment**

6.2.12 FrameComponent Class

Method	%	Rationale
getSignature()	25%	The AICDM can model a very limited range of signature types.

25%	**FrameComponent class Degree of Alignment**

6.2.13 Platform Standard Object Degree of Alignment

The degree of alignment of the Platform standard object to the AICDM is the average of the degrees of alignment of its classes:

$$\frac{62+30+50+25+75+75+25+100+81+85+25+25}{12} = 55\%$$

The error in the degree of alignment is:

$$\frac{4+13\times\left(6.25/\sqrt{3}\right)+21\times\left(12.5/\sqrt{3}\right)}{35} \approx 6$$

(4 represents the error from the Location standard object. The Platform standard object contains 13 methods whose degrees of alignment are assigned either 0 or 100 and therefore have a standard deviation of $6.25/\sqrt{3}$. The other 21 methods have a standard deviation of $12.5/\sqrt{3}$.)

6.3 Location Standard Object

6.3.1 Location Class

Method	%	Rationale
distanceFrom()	25%	Two factors determine the degree of alignment: • The AICDM models geodetic but not Cartesian coordinates. • The AICDM can model the method's inputs (two locations) but not its result; that is, the AICDM does not model distances.
convert()	50%	The AICDM models geodetic but not Cartesian coordinates. In other words, for a given invocation, it can model the method's input but not the output, or vice versa.[8]

37%	**Location class Degree of Alignment**

6.3.2 Local Class

Method	%	Rationale
getXCoordinate()	0%	The AICDM does not model local coordinate systems.
getYCoordinate()	0%	
getZCoordinate()	0%	

0%	**Local class Degree of Alignment**

6.3.3 LatLon Class

Method	%	Rationale
getLatitude()	75%	The precision for latitudes and longitudes differs between the AICDM and the OMSC.
getLongitude()	75%	
getAltitude()	75%	The AICDM can specify altitudes to varying degrees of precision. Moreover, the AICDM uses the metric system, and the OMSC uses the English system. This can cause some minor errors in conversion.

75%	**LatLon class Degree of Alignment**

[8] Many simulation systems convert between Cartesian and geodetic coordinates. Arguably, then, the degree of alignment for convert() should be higher. However, there are many competing models for converting between coordinate systems, depending on such factors as line-of-sight computation needs. The AICDM and the OMSC are intended as domain standards. Since standards in coordinate conversion have not emerged, it seems safest to assume coordinate conversion can't be automated.

6.3.4 Location Standard Object Degree of Alignment

The degree of alignment of the Location standard object to the AICDM is the average of the degrees of alignment of its classes:

$$\frac{37+0+75}{3} = 37\%$$

The error in the degree of alignment is:

$$\frac{3\times\left(6.25/\sqrt{3}\right)+5\times\left(12.5/\sqrt{3}\right)}{8} \approx 4$$

(The Location standard object contains 3 methods whose degrees of alignment are assigned either 0 or 100 and therefore have a standard deviation of $6.25/\sqrt{3}$. The other 5 methods have a standard deviation of $12.5/\sqrt{3}$.)

7. Recommendations

The analysis has led to recommendations on what must be done if the AICDM and the OMSC are to be brought into alignment. These are recommendations for changes to the AICDM and to the OMSC.

Some of the recommendations are straightforward and narrow in scope, affecting a small portion of one or the other model. (An example would be to add a value to an AICDM attribute's domain.) These recommendations are enumerated in Section 7.1.

A few of the recommendations are broader. They argue for a philosophical shift to a model. They claim that in such things as the type of entities modeled or the nature of how entities are modeled, either the AICDM or the OMSC (or perhaps both) are incomplete as regards alignment. A recommendation of this sort clearly needs justification. Section 7.2 presents the broader set of recommendations, and argues for their importance in achieving alignment.

7.1 Specific Recommendations

This section presents specific recommendations for changes to the AICDM and the OMSC. Each change is intended to promote alignment. The materiel is organized according to standard objects, classes within standard objects, and methods within classes. For each method, a table presents issues driving the recommendations, and the recommendations themselves.

The recommendations for the AICDM and the OMSC are split into separate sections to promote an understanding of the changes suggested for each method. Appendix D presents the changes grouped by standard objects, classes within standard objects, and methods within classes. For each method, a table presents issues driving the recommendations, and the recommendations themselves. This helps the reader see how alignment issues jointly drive changes to each model.

The material in this section is intended as preliminary. Further study is needed to determine the utility and form of each recommendation.

7.1.1 Recommended Changes to the AICDM

Method	Issues	Recommendations
Unit Standard Object		
The Unit Class		
getVelocity()	The AICDM does not model velocity.	The AICDM needs to be able to model the velocity of (at least) the PERSON, MATERIEL, and ORGANIZATION entities.

Method	Issues	Recommendations
getSide()	The AICDM does not have an explicit model of sides, although one can be simulated using ORGANIZATION structures.	The AICDM should enhance its ability to model friends and foes. One approach is to add values FACTION and COALITION to the IDENTIFICATION-FRIEND-FOE CODE attribute of the ORGANIZATION entity. Another is to externalize the FRIEND-FOE attributes.
getPosture()	The AICDM has a limited set of attributes that can represent posture. Their values do not cover the examples of status given in the OMSC.	The AICDM needs to add an attribute associated with an ORGANIZATION that can represent posture. Since different types of organizations can have different types of posture (e.g., a civilian organization will not have the same types of a posture as a military organization), it might make sense to have multiple posture attributes, each associated with a subtype of ORGANIZATION.
getStatus()	The AICDM has a very limited set of attributes that can represent status. Their values do not cover the examples of status given in the OMSC.	Necessary changes to the AICDM cannot be determined until the concept of status in M&S systems is better understood. However, it is likely that: • The AICDM will need to accommodate numeric descriptions of status (e.g., as percent effectiveness). • The AICDM will need additional codes.
The UnitGeometry Class		
getOrientation()	The AICDM does not model orientation.	The AICDM should include information on orientation. It is likely that orientation must be specified for several types of entities (organizations and platforms, at least). Given the attributes of POINT, which include precision, it is probably necessary to create a new entity, ORIENTATION, containing attributes that express orientation. Instances of this entity would be linked to an ORGANIZATION (or PLATFORM) through a many-to-many relationship via an intermediate ORGANIZATION-ORIENTATION entity (similar to the relationship between ORGANIZATION and LOCATION). An orientation may be absolute or relative (e.g., “in front of”), and the AICDM should be able to model both types.

Method	Issues	Recommendations
The Supply Class		
• getRemainingCapacity() • getTotalCapacity() • transfer()	The AICDM does not model capacity	The AICDM needs entities that model capacity.
The Maintenance Class		
conductMaintenance()	The AICDM does not have an attribute of ACTION that can record maintenance.	A new value, CONDUCT MAINTENANCE, needs to be added to the ACTION-VERB CODE attribute of ACTION.
The Intel Class		
collect()	The AICDM does not model sensor activation and deactivation.	The AICDM needs attributes that give the state of a sensor, as well as other possible sensor-specific characteristics (e.g., orientation)
reportContacts()	Typically, M&S systems can model any entity detected by intelligence. The AICDM can only record such an entity if it is a candidate target.	• The AICDM needs to model entities recorded by intelligence but not yet known to be (candidate) targets. • The codes associated with FEATURE need to account for intelligence. • The proposed AICDM entity INFORMATION-REFERENCE may be useful for modeling intelligence results. The domain values of the INFORMATION-REFERENCE-CATEGORY-CODE attribute would need to be extended to account for intelligence.
Platform Standard Object		
The Platform Class		
getType()		For convenience, the AICDM may need to be amended to include a new entity class PLATFORM. Subtypes of this entity would include all AICDM entities that are considered platforms in The OMSC. (The DDA contains separate entities for ships and satellites. We recommend including these entities in the AICDM to address M&S requirements for water and space platforms.)
assessDamage()	The AICDM codes for damage to materiel, facilities, and people are not adequate to model the types of damage possible in an M&S system.	The AICDM needs codes for materiel, facilities, and people that model the types of damage possible to such entities in M&S systems. Possibly some types of damage are complex enough to warrant creation of a new entity, with attributes to model the various types of damage.
The Sensor Class		
• activate() • deactivate()	The AICDM does not model whether a sensor is on or off.	The AICDM needs to model sensor state.

Method	Issues	Recommendations
The Weapon Class		
load()	The AICDM does not model whether or not a weapon is loaded.	The AICDM needs to be able to model the state of a weapon.
engageTarget()	The AICDM does not model weapon firing.	The AICDM needs to model weapon firing. More precisely, it needs to model the status of a weapon, including being in a firing state (whatever that might mean for a particular weapon).
The PlatformFrame Class		
getSignature()	The AICDM can model only a cross-sectional area signature.	The AICDM should increase the range of signatures that it can model. (The DDA has attributes to keep track of height, weight, etc., that the AICDM lacks.)

7.1.2 Recommended Changes to the OMSC

Method	Issue	Recommendations
The Unit Standard Object		
The Unit Class		
getLocation()	• The OMSC is ambiguous about how a unit location is modeled. It says: Typically [current unit location] is the center of mass or some other point location representative of the unit location. If center of mass is typical, there must be some atypical representations. The OMSC does not enumerate them. • The OMSC does not define how center of mass is computed. M&S applications use different models that depend upon factors such as level of aggregation (corps/division vs. theater) and whether the simulation is for training or analytical purposes. (In some M&S training applications, center of mass is not computed at all, but is estimated by (and entered by) a human operator.)	• The OMSC should enumerate all valid ways in which a unit's location can be modeled. A better approach would be to accept center of mass as the standard way to represent location. Other location-like quantities (e.g., areas) should be represented using other classes. • The Location class should have a virtual method centerOfMass(). An M&S application based on the OMSC would need to supply a definition for the method.
getVelocity()	The OMSC's description does not prescribe the units in which velocity components are expressed.	The OMSC should define a (parameterized) model of velocity. The model must describe units and degree of precision (or these quantities must be parameters).
getID()	The OMSC does not indicate how the ID might be used. Use can prescribe format. (E.g., if an ID labels units on a screen, it must be short.) This influences whether the AICDM can model OMSC IDs using existing attributes.	The OMSC should define a model of IDs based on their use in existing M&S systems. Ideally, the OMSC should use C4I IDs.

Method	Issue	Recommendations
getSide()	• The OMSC does not indicate the maximum number of sides that must be supported. At one time it was assumed that the only sides would be "blue" and "red", but with the advent of coalition warfare the number of possible sides has grown. Note that WARSIM has a contractual requirement to support at least 36 different sides. • The OMSC states that there is no implied enmity between sides, but does not state the possible relationships between sides.	• The OMSC should state explicitly whether there is an implied maximum number of sides. • The OMSC should enumerate the possible relationships between sides.
getPosture()	• The OMSC does not enumerate all possible postures. • The OMSC does not relate postures to other objects. For instance, a posture such as "hasty defense" implies that a unit's velocity is zero.	• The OMSC needs to clarify the role that postures play in simulations and the possible values they may have. Do they need to be distinguished from current activities, as elaborated by the current activity codes of the ORGANIZATION and MILITARY-UNIT entities? • Whatever a posture may be, the set of valid postures for a given M&S system can apparently be expressed as an enumeration. This suggests what the result of *u*.getPosture() should be for some unit *u*.
getStatus()	The OMSC does not enumerate the different types of status.	The OMSC should define a virtual class *Status*. Subclasses of *Status* would exist for every entity that needs to record status. Subclass methods would record and provide different facets of status as appropriate to the entity.
getMission()	The OMSC gives no structure to a mission or task. A mission is only free text. The AICDM, by contrast, has a rich structure that the OMSC cannot model.	The OMSC should define a Mission standard object that provides a more precise specification of what tasks are and how they will be used.

Method	Issue	Recommendations

move()	• The OMSC does not state the parameters of the method. Potential parameters include:
	  • New coordinates (either relative or absolute)
	  • Time until new coordinates are reached
	  • The specification does not state whether the next location is known in advance (in which case parameters might not be needed).
	• The OMSC does not describe necessary granularity of movement (that is, minimum length or time of movement).
		The OMSC should included a parameterized model of movement, one that addresses granularity.

determineAttrition()	The OMSC does not specify units for attrition (percentage? absolute values? specific entities removed?)	The OMSC should specify ways in which attrition may be stated. Where applicable, the OMSC should specify units and precision for attrition.
The UnitGeometry Class		
getShape()	The OMSC does not indicate the nature or generality of bounding shapes.	The OMSC should define the result of getShape() to be a class *Shape*. *Shape* should have subclasses such as *Rectangle*, *Circle*, etc. This structure will support current needs while providing for future enhancements.
getOrientation()	The OMSC does not state the units of orientation. Presumably orientation is in degrees.	The OMSC should state the units and precision of orientation.
The Supply Class		
• getRemainingCapacity() • getTotalCapacity() • transfer()	The OMSC does not prescribe how capacity is expressed.	The OMSC needs a standard model, or perhaps set of models, for expressing capacity.
The Maintenance Class		
conductMaintenance()	• The OMSC does not specify how maintenance actions or medical treatment are stated. It does not specify resultant states. • The OMSC does not specify if maintenance is performed on individual items or on groups of items.	The OMSC needs a more exact model of maintenance. It must be possible to specify the type of maintenance to be performed, the expected result, the materials and effort consumed during maintenance, etc.

Method	Issue	Recommendations
conductRecovery()	The OMSC describes recovery as if it were an all-or-nothing action: recover all entities. Presumably an M&S system can be more selective, opting to perform triage (for example).	The OMSC needs a model of recovery that describes: • How to determine what to recover. • The resources consumed during recovery.
conductEvacuation()	• The OMSC does not state how the location of a rear area is determined. • The OMSC does not define if evacuation must be performed en masse. Must a whole unit be evacuated, or can portions be evacuated?	• The OMSC should include a location as a parameter to the method. • The OMSC needs a model of evacuation that, like recovery, precisely defines what recovery is and the resources it consumes.
The C2 Class		
doC2()	The OMSC does not describe the nature of command decisions or control actions.	The OMSC should create a set of classes that can model C^2 actions important to for M&S.
The Attrition Class		
causeAttrition()	• The OMSC does not define the actions one unit can take that might cause losses to another unit. If those actions include firing weapons, it would seem more logical to invoke the weapon firing method directly than to invoke it through this method. • The OMSC does not specify how much attrition is caused by invoking this method.	The OMSC needs a model that relates attrition to the various actions one unit might take against another.

Method	Issue	Recommendations
The Communications Class		
• getNet() • setNet()	• The OMSC does not specify the types of objects that might be modeled in a network. Presumably it includes components of the unit hierarchy. It may also include platforms. • Does "Capable of exchanging messages" include both friends and foes? • Is the intent of communications to capture electronic message exchange? Or does it include other types of signals (e.g., semaphores, written communication, or for that matter verbal communication)?	The OMSC needs a more precise model that captures how units communicate in simulations
The Intel Class		
collect()	• The OMSC does not specify how the organic sensor assets of a unit are defined. • Detection involves more than turning on a sensor. The sensor is typically directed against some feature, other organization, etc. The OMSC does not define how search capabilities may be effected other than turning them on.	• The OMSC must specify how the sensor assets of a unit are defined. • The OMSC must specify how the sensor assets of a unit may be used.
reportContacts()	The OMSC does not specify the nature of results, nor how they might be used. The only other method associated with a Unit that might use intelligence is C2.doC2().	The OMSC needs a class that acts as a superclass of possible results for the collect() method.

Method	Issue	Recommendations
The Platform Standard Object		
The Platform Class		
getType()	The OMSC does not define what a type designation is. Assuming that Platform is a virtual class and that actual platform instances would be members of subtypes of Platform, the type designation is probably just an identifying string that gives the name of the subtype.	The OMSC needs to enumerate at least a partial list of valid platform types.
assessDamage()	• The OMSC does not specify the types of objects that might cause damage. • The OMSC does not specify how the amount of damage is determined. • The description states that the method calculates damage caused, but there is no method associated with a Platform that actually causes damage. • The OMSC does not specify units in which damage is measured. • The OMSC does not specify how the object that caused the damage is identified, if more than one object can cause damage, or what an object might be (Can damage to a truck come from a rock? If not, how does one model such potential damage?). • This method assesses damage. How is damage caused? What entities are consumed in causing it?	The OMSC needs a model of damage.
The Sensor Class		
getMaxRange()	The OMSC does not prescribe the units of range.	The OMSC should standardize the units and precision of range.

Method	Issue	Recommendations
getContacts()	A Platform does not have a location. Without a location, how can the contacts be determined? (The Unit class has a collection of Platform instances, and moreover may be hierarchically composed of Unit instances. Perhaps a Platform's location is always the center of mass of its containing Unit, where the Unit's granularity is small enough to resolve the problem of locating contacts.)	The OMSC needs to clarify the method by which contacts are gathered, and whether it requires a location; if it does, The OMSC needs to clarify how the location of a platform is determined.
The Weapon Class		
getMaxRange()	The OMSC does not prescribe the units of range.	The OMSC needs to define the units and precision of range.
load()	The OMSC's description of loading "a" munition seems oriented towards particular types of weapons, like artillery. Is that the intent?	The OMSC needs to clarify the range of weapons to which this operation can be applied.
engageTarget()	• The OMSC does not indicate how long the weapon-firing event takes. Is this captured in a subtype? It should be a virtual method. • There is no method to determine whether weapon firing has ended. • Is aiming part of engaging a target? There is no aim() method.	The OMSC needs a model of weapon firing. The model should allow a system to determine if a weapon has been fired, where it's pointing, and things like that.
The PlatformFrame Class		
getSignature()	• The OMSC lists examples of, but does not describe the full range of, signatures. Nor does it provide a general means to encapsulate signatures (e.g., a virtual class *Signature*). • The method's description is worded as if a target has a single signature. Can't a target have multiple signatures?	The OMSC should define a virtual class *Signature*, which would be the value returned by getSignature(). Sensors would return subclasses (ThermalSignature, AcousticSignature, etc.).

7.2 General Recommendations

7.2.1 Recommendations for the AICDM

The AICDM is a comprehensive model of C2 systems. Its designers did not, however, consider modeling and simulation system entities when they designed it. The following subsections list general changes to the AICDM that would improve its ability to serve as a data model for M&S systems.

7.2.1.1 Entity State

Knowing the current state of an entity is vital in a simulation. Much of the work done by an M&S system is concerned with computing a future state, and a future state is determined by the current state. The AICDM records certain state attributes, but mainly those of interest in the logistics community. Thus, the AICDM models a vehicle's location, but does not model its velocity (either speed or orientation[9]). But using velocity as the basis for calculating a future state of a vehicle is one of the most elementary operations of simulation.

We therefore recommend that the AICDM be augmented to include the capability to model a set of entity states deemed of interest to the M&S community. The following properties must be added to support current OMSC methods:

- Velocity
- Sensor state (on or off)
- Weapon state (loaded vs. unloaded, firing vs. idle)

7.2.1.2 Physical Entity Properties

M&S systems need to know certain static properties of the entities they model. The OMSC, for example, makes use of platform signatures as part of intelligence collection.

The AICDM models a few static properties, in particular size and weight. But again, it sticks to properties of interest to the logistics community. M&S systems need thermal and electronic signatures, among others.

We recommend that research be conducted on modeling entity properties, and that a data model for these properties be incorporated into the AICDM.

7.2.1.3 Joint, Foreign, and Commercial Entities

Currently, the AICDM is focused on meeting the Army warfighter's data needs for C4I systems and includes extensive representations for Army forces (military units) and their materiel. However, in training simulations, many other entities (organizations, vehicles, equipment, facilities) besides those under U.S. Army Command and Control may need representation in order to represent the interactions of U.S. Army forces with Joint and Coalition Forces, as well as opposing forces and civilian entities. Hence, if the AICDM is

[9] Note too that the orientation component of velocity may not be enough information for a simulation.C4I modeling sometimes assumes that an object is oriented in the direction it's moving. An aircraft in a crosswind violates this assumption.

to support these M&S data representation requirements, it would benefit from an expansion of its entities and relations to cover more types of entities outside of the current focus on U.S. Army entities.

Some more specific requirements include entities for water vehicles and spacecraft to accommodate interactions with naval vessels as well as possible communications dependencies on satellites. Supporting just these requirements would improve AICDM modeling capabilities for Joint exercises. It appears as though the adequate representation of foreign forces may require other additions as well. The current AICDM representations for MATERIEL, for example, seem to have dependencies on materiel-item identifiers and national stock numbers which may not be available for the equipment of many foreign forces. This issue requires further investigation.

Modeling of civilian and commercial entities could benefit from a richer hierarchy of organization types. Currently, the only subtypes of the AICDM ORGANIZATION entity are the UNIFORMED-SERVICE-ORGANIZATIONs of CONVOY, MILITARY-UNIT, and ORGANIZATION-POST. Civilian organizations may benefit from additional subtypes to capture their relevant characteristics, especially the so-called Non-Governmental-Organizations (NGO's) which interact with the military in Operations Other Than War (OOTW) such as disaster relief. Further investigations are required to assess the extent to which additional data representations are needed for foreign and civilian entities to support M&S requirements.

Existing parts of the DDA which are outside of the current scope of the AICDM can satisfy most of these requirements. The DDA SHIP entity and its rich web of associated entities, for example, should meet Army M&S requirements for water vehicle representations. There is also a SATELLITE entity in the DDA which may meet M&S requirements for representing spacecraft.

There are a variety of alternative approaches to meeting these new data representation requirements. The AICDM could incorporate appropriate existing entities from the DDA and add new ones as required. Separate "annexes" could be created for the AICDM for those data requirements that do not meet C4I core data needs, thereby keeping the AICDM itself as a relatively streamlined "core" for C4I data modeling. Alternatively, applications requiring these extended capabilities might be referred to the general DDA, or parts thereof, as the source for non-core data representations. We do not make any general recommendations on a preferred course of action here, as broader policy issues are relevant. But, we do recommend that a policy be developed for handling representations of such data, when it falls outside of the "core" scope of Army C4I systems.

7.2.1.4 Linkages Between Existing Entities

While the AICDM generally benefits from a very rich set of associations amongst its entities, there is at least one case in which important relationships do not exits. The example we've identified of this is for the AIRCRAFT entity, which would ordinarily be considered a type of materiel, along with other vehicles, such as tanks. However, the AIRCRAFT entity does not have any connection to the MATERIEL entity in the examined AICDM model. As a result, it cannot use MATERIEL entity associations to identify its associated location, crew, or materiel as supplies or cargo. Since the AIRCRAFT entity is found in an area of the

AICDM that is still "to be worked," this deficiency may will be rectified by the time these entities are ready for submission to DISA in a proposal package. Also, the DDA contains extensive additional AIRCRAFT-related entities which could be incorporated in the AICDM to meet these requirements.

7.2.1.5 Non-Administrative Organizations

The AICDM ORGANIZATION entity models "an administrative structure with a mission." A strict interpretation of this phrase would preclude the AICDM from modeling operational structures—e.g., ad hoc organizations, such as a group of wounded warfighters being evacuated. An operational structure entity would be useful in modeling the effects of The OMSC methods that operate on quantities of items.

The AICDM should clarify its intention regarding the possible use of ORGANIZATION to model operational as well as administrative structures. If ORGANIZATION is to be used solely for administrative structures, the AICDM should be enhanced to include entities and relationships that can model operational structures.

7.2.2 Recommendations for the OMSC

As a high-level architectural model of M&S systems, the OMSC has goals not connected to data model alignment. Nevertheless, there are several pervasive but straightforward changes that could be made to the OMSC in order to improve alignment. Appendix B motivates the need for these changes. The following subsections describe them.

7.2.2.1 Resolve Ambiguities

The OMSC's deliberately vague descriptions of its classes and methods prevents real understanding of the degree to which the OMSC and the AICDM align. The OMSC should address this situation by taking the following actions:

- ***Provide parameter schemata for all methods.*** This would provide useful detail on all entities associated with an action. Movement is an example that has been mentioned throughout this report. Another good example is attrition. If the OMSC defined the parameters of the causeAttrition() method, it would be possible to determine whether the AICDM models the entities that cause attrition, the amount of attrition they can cause, the time at which they cause it, etc.

It has been pointed out that not specifying a method's parameter scheme allows more forms of a method to fit into the general architecture. In that case, the architecture becomes more notional than concrete. The AICDM is a concrete data model, and aligning it to a notional M&S model is difficult, as this report has shown. In any case, overloading can be used to accommodate multiple parameter schemata.

- ***Specify return values for all methods.*** This action is analogous to the previous one. Specifying a method's return value better describes the types of entities with which a method interacts.
- ***Include class constructors.*** Constructors help establish what constitutes a valid instance of a class. (For example, must a unit have an ID?) If the OMSC provided constructors, we could map the action of creating an object instance to a set of AICDM entities, attributes, and relationships. Currently we can define the

necessity of attributes having values, and of relationships existing between entities, based on invoking a method, but we cannot aver that a particular set of relationships must always exist, or that a specific set of attributes have interrelated values.

- ***Include set as well as get methods.*** This action is analogous to including class constructors. Set methods often make apparent constraints between instance properties. They also show which properties are immutable, and which cannot be directly defined (e.g., a setLocation() method is unlikely—location must always be changed using the move() method). These extra semantics facilitate alignment.

UML provides the opportunity to capture this and much more information.

The OMSC should also consider using an established documentation paradigm, such as that used by Sun Microsystems to specify Java classes,[10] to ensure that each standard object is described as fully as available information permits. Java documentation succinctly captures the interface of a class. It is not sufficient—it does not fully define a class' semantics—but it presents considerably more information than the OMSC.

7.2.2.2 Base Class Designs on Objects, Not Algorithm Encapsulation

The OMSC's Unit and Platform standard objects consist of a focal class and an aggregation of other classes. Almost all of these aggregated classes are designed to encapsulate algorithms rather than to model objects.

For example, the Unit standard object's Attrition class has a method that causes attrition. It is difficult to conceive of attrition as an object. There is no physical attrition entity. Attrition is a concept to describe loss within a unit. It is possible that a simulation system might implement a message passed between units that describes attrition. But describing such a message seems out of place in the OMSC's class model. It belongs in a process model.

The AICDM models entities. Because attrition is not an entity, it does not exactly align to anything in the AICDM. This report has speculated on the characteristics of attrition and attempted to find AICDM entities that can suffer attrition, with attributes describing how. The discussion of determineAttrition() and causeAttrition() is the weaker for it. The OMSC's class model should instead concentrate on identifying and modeling objects.

7.3 Recommendation for Future C4I-M&S Alignment Studies

Finally, we offer a recommendation on addressing the broader issue of alignment of C4I and M&S data models. The purpose of this recommendation is to promote studies that optimize the benefits of alignment to both the C4I and the M&S communities.

The recommendation is to avoid a mismatch of models such as occurred between the AICDM and the OMSC. The alignment analysis presented here was severely limited by the disparity between the modeling levels of the OMSC and the AICDM. This report pro-

[10] See http://java.sun.com/j2se/1.3/docs/api/index.html for examples of the Java documentation format.

vides a good framework for beginning actual alignment of an OMSC-based system to an AICDM-based system. But the results are based on numerous hypotheses identified in Appendix A, and any actual alignment is likely to require work in areas not foreseen in this report. More plainly, some of the methods' alignment ratings will prove too high because of false assumptions.

In the future, the AICDM should be aligned to an M&S model that provides much more detail on classes, methods, and semantics.

References

[AMSAA] OMSC Overview.http://www.amsaa.army.mil/objects/docs/overview1.pdf.

[Army 2000] Joint Technical Architecture – Army, Version 6.0 (May 2000). See http://arch-odisc4.army.mil/aes/aea/jta-a/jtaa60/html/jtaa60.htm.

[Booch 1996] Grady Booch, *Best of Booch: Designing Strategies for Object Technology,* Cambridge University Press, New York, New York, 1996.

[DoD 1991] Dod-8320.1, *Data Standardization Procedures,* September 1991. See http://www-datadmn.itsi.disa.mil/guidance.html.

[DoD 1998] DoD-8320.1-M-1, *Data Standardization Procedures,* April 1998. See http://www-datadmn.itsi.disa.mil/8320_1m1.html.

[DoD 1999] Department of Defense Joint Technical Architecture, Version 3.0, (November 1999). See http://www-jta.itsi.disa.mil/.

[H 2000] Don Hodge, "Comments on The Alignment Of Army Integrated Core Data Model and Object Management Standards Category Study", private communication.

[HB 1998A] Don Hodge and Brad Bradley, *Army Standard Unit Object*, TR-638, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, MD (October 1998).

[HB 1998B] Don Hodge and Brad Bradley, *Army Standard Platform Object,* TR-634, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, MD (July 1998).

[HB 1999] Michael Hieb and James Blalock, "Data Alignment between Army C4I Databases and Army Simulations," Paper 99S-SIW-034, 1999 Spring Simulation Interoperability Workshop (March 1999).

[HLA 2000] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules, IEEE, Piscataway, New Jersey (September 2000). See http://www.dmso.mil/index.php?page=121.

[HS 2000] Michael Hieb and Ron Sprinkle, "Simulation Infrastructure for the DII COE Architecture: The Army Vision," Simulation Interoperability Workshop, Orlando, FL (September 2000).

[JHB 1998] Leroy Jackson, Don Hodge and Brad Bradley, *Army Standard Location Object,* US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, MD (October 1998).

[LSCZ 1998] Michael Lightner, J. Schanduaa, D. Cutts, and S. Zeswitz, "The High Level Architecture Command and Control Experiment – Lessons Learned in Designing an Extended Federation", Paper 98S-SIW-93, 1998 Spring Simulation Interoperability Workshop (March 1998).

[NIST 1993] *Information definition for information modeling (IDEF1X),* National Institute of Standards and Technology, Gaithersburg, MD, 1993 (FIPS Publication 184).

[SNF] Reference 007-97, *Surveying of Navigation Facilities,* 1997. http://www.wgs84.com/files/stanwgs8.pdf.

Abbreviations and Acronyms

ABCS	Army Battle Command Systems
AFATDS	Advanced Field Artillery Tactical Data System
AICDM	Army Integrated Core Data Model
AMSAA	Army Materiel Systems Analysis Activity
AMSO	Army Model and Simulation Office
C2	Command and Control
C2CDM	C2 Core Data Model
C3I	Command, Control, Communications & Intelligence
C4	Command, Control, Communications, and Computers
C4I	Command, Control, Communications, Computers, & Intelligence
COE	Common Operating Environment
DB	Database
DBMS	Database Management System
DDA	DoD Data Architecture
DDDS	Defense Data Dictionary System
DDM	DoD Data Model
DII	Defense Information Infrastructure
DISA	Defense Information Systems Agency
DoD	Department of Defense
ER	Entity-Relationship
FOM	Federation Object Model
GH4	Generic Hub data model, version 4
HLA	High Level Architecture
IDA	Institute for Defense Analyses
IDEF1X	Integrated Definition for Information Modeling
IEEE	Institute for Electrical and Electronics Engineers
JCDB	Joint Common Database
JTA	Joint Technical Architecture
JTA-A	Joint Technical Architecture—Army
M&S	Modeling and Simulation
MIDB	Modernized Intelligence Database
MRCI	Modular Reconfigurable C4I Interface
NGO	Non-Governmental Organization
NIST	National Institute of Standards and Technology
ODISC4	Office of the Director for Information Services for C4

OMSC	Object Management Standards Category
OO	Object Oriented
OOTW	Operations Other Than War
SQL	Structured Query Language
URL	Uniform Resource Locator
UML	Unified Modeling Language
WARSIM	Warfighters Simulation

Appendix A.
Alignment Analysis

This appendix presents the alignment analysis. It identifies how—or if—AICDM entities and attributes relate to OMSC standard objects, classes, and methods. It provides a reference guide to which concepts, entities, and properties align between the two models. It does not define degree of alignment, but instead gives the preparatory work needed for calculating degree.

Appendix A is organized hierarchically, as shown in Figure A-1. Nodes in the tree represent sections. Top-level sections correspond to Conceptual level alignment. Subsections correspond to Entity level alignment. Sub-subsections correspond to State level alignment.

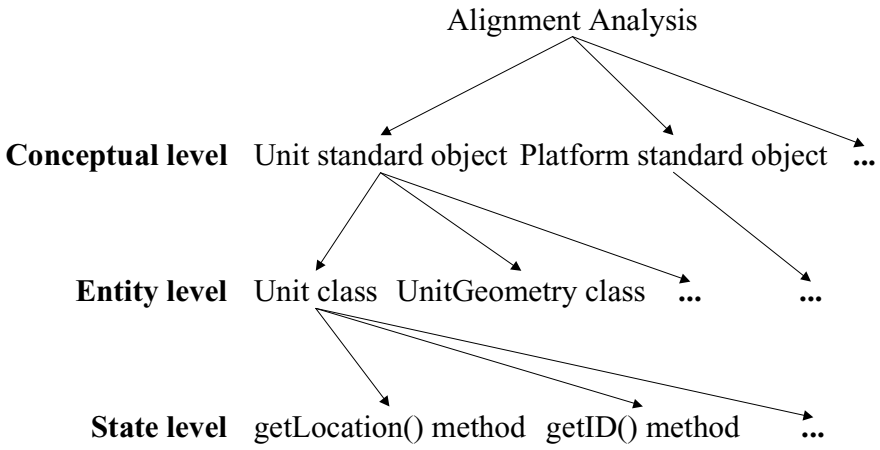


Figure A-1. Organization of Alignment Analysis

The analysis is organized around OMSC modeling elements, reflecting the focus of our alignment on mapping from OMSC to AICDM elements (see Section 5). Thus Conceptual level analysis begins by choosing an OMSC standard object and identifying or defining an AICDM view that aligns with it. Entity level analysis aligns each OMSC class within a standard object with AICDM entities from the corresponding AICDM view. State level analysis consists of examining a class' methods and, for each method, identifying corresponding AICDM attributes within entities that align to the class.

There is no Value level alignment. The OMSC does not provide enough details to support it. The definition of alignment gives attribute domains no role outside of Value level alignment. But sometimes an examination of an AICDM attribute's domain helps support

State level analysis. There is no point in discarding useful information, so it's captured. Paragraphs containing information that properly belongs at the value level are specially marked:

> A paragraph containing information derived from Value level analysis has a tag at the top of its right margin. VALUE

The analysis results for the Conceptual level appears in tables. The following is a brief explanation of these tables.

For each OMSC standard object, there is one table containing Conceptual level analysis. This table has three columns:

- ***OMSC Class:*** This column lists all the classes in a given standard object. Each row lists one of the classes from the OMSC standard object.

  As Section A.2 explains, there are four categories of platforms. The Conceptual level analysis table for the Platform standard object (page A-49) accordingly breaks classes into four rows, one for each platform category.

- ***Related AICDM Entity:*** This column lists the major AICDM entities in the view that align with the OMSC classes in the first column (other entities enter the view through relationships; see the last bullet). Each row in the column lists one or more entities that have some degree of conceptual alignment with the OMSC class in the first column of that row.

  If an AICDM entity is a subtype in a hierarchy, its supertypes are only listed in this column the first time the entity appears. Supertypes can be identified via the relationships traced to the focal entity in the third column.

- ***Relation to Focal Entity:*** Conceptual level alignment specifies not only AICDM entities but how those entities relate to the focal entity of the view. This column defines that relationship for the entities in the first column. The definition is in terms of figures, expressed in the IDEF1X notation. The reader can refer to the reference figure to determine the exact set of entities and relationships.

  Entities in the first column aren't always associated with the focal entity by a single relationship. For one thing, if two entities have a many-to-many relationship, the AICDM expresses that relationship using an intermediate entity (e.g., MATERIEL-LOCATION relates MATERIEL to LOCATION). For another, an entity in the first column is often a subtype of some other entity, and the latter entity is the one that participates in relationships (e.g., MILITARY-UNIT is a subtype of ORGANIZATION; ORGANIZATION, not MILITARY-UNIT, is related to PERSON, MATERIEL, etc.). The AICDM view that is aligned with a standard object includes the intermediate entities and supertypes, as listed in the third column. The view therefore consists of all entities and relationships in the second column of the table, plus all entities and relationships implied by the third column of the table.

The analysis results for the Entity level appears in a table if the OMSC class has methods. Otherwise, it is stated as free text. For each OMSC class with methods, there is one Entity level analysis table. This table has the following three columns:

- ***AICDM Entity:*** This column captures both those entities identified as aligned with an OMSC class at the Conceptual level and any additional AICDM entities that cover the specific representational needs of the methods of that OMSC class. The set of entity names in each row aligns to the OMSC class in question.

- ***Suggested By:*** This column contains the names of methods of the table's OMSC class. Each row has one or more methods of the class being analyzed. Each method's name motivates the need for the entity (or entities) in the first column.

- ***Relation to Focal Entity:*** Entity level alignment specifies not only AICDM entities but how those entities relate to the focal entity of the view. This column defines that relationship for the entities in the first column.

  The intermediate entities listed in this column aren't in the set of entities that align to the OMSC class. A standard object specifies a relationship between its focal class and other classes (either aggregation or inheritance). This column demonstrates that the AICDM can model that relationship.

The analysis results for the State level are stated as free text. Most OMSC methods have ambiguities. All ambiguities that hinder State level alignment analysis are noted. The analysis then presents, using the standard phrasing from Table 7 (reproduced below as Table A-1), the estimated degree of alignment for the method. (The phrase is shown in boldface type for easy identification.) This is followed by detailed discussion justifying the estimate.

Table A-1. Possible Degrees of Alignment

Value	Standard Phrase	Meaning
0%	No alignment	This value is assigned in either of the following circumstances: • There is no overlap between the models. One model contains an instance of an element that has no analog in the other. • Lack of information in the OMSC completely prevents alignment analysis.
25%	Low degree of alignment	There is some overlap, but it seems coincidental. Overlap might have been achieved by using AICDM attributes in ways that its designers did not originally intend.
50%	Medium degree of alignment	There is a moderate amount of overlap, but still a significant disconnect between the models.
75%	High degree of alignment	Perfect alignment can probably be achieved by small changes to one model or the other.
100%	Perfect alignment	There is an exact, unambiguous mapping between the models.

The tables that present Conceptual and Entity level alignment should be regarded as summaries. They do not attempt to provide all the details on the AICDM and the OMSC, or on how the models are used. Notes are often used to explicate material in the tables, and figures are provided to show AICDM entities, attributes, and relationships (consistent with the definition of alignment, figures in the sections on Conceptual level alignment usually show entities without attributes, and figures in the sections on Entity and State level alignment usually show entities with attributes). Also, the lowest level subsections have some free text descriptions of subtle points. In any case, it is probably wise to read this appendix with documentation for the AICDM and the OMSC standard objects near at hand.

A.1 Unit Standard Object (Conceptual Level)

The OMSC defines Unit as follows [HB 1998A]:

> A Unit encompasses military organizations that represent collections of entities (e.g., people, vehicles, weapon systems, etc.). Examples of this definition include organizations (i.e., companies, battalions, brigades, divisions, etc.) as well as functional groups (e.g., Tactical Operations Centers and Fire Control Centers).

The class Unit is the focal class of the Unit standard object.

The AICDM ORGANIZATION view contains the principal entities that model military organizations. The MILITARY-UNIT entity in the ORGANIZATION view aligns most closely with the OMSC's Unit class and is used as the focal entity of our corresponding AICDM view.[1] Table A-2 shows the entities of interest, and how they relate to a MILITARY-UNIT.

[1] Arguably, the FACILITY entity aligns most closely with a fire control center. If it is appropriate to model a fire control center as a FACILITY, that FACILITY can still be related to an ORGANIZATION through the ORGANIZATION-FACILITY entity.

Table A-2. AICDM Entities that Align with Classes of the Unit Standard Object at the Conceptual Level

OMSC Class	Related AICDM Entity	Relation to MILITARY-UNIT
Unit	MILITARY-UNIT	Identical (the same entity)
UnitGeometry	SURFACE and its subtypes	A MILITARY-UNIT is a subtype of a UNIFORMED-SERVICE-ORGANIZATION, which is a subtype of ORGANIZATION.[2] An ORGANIZATION has an associated CONTROL-FEATURE (a subtype of FEATURE) that is used to describe the organization's shape. The AICDM models the shape through an association between a CONTROL-FEATURE and a LOCATION; one of the subtypes of LOCATION is SURFACE, whose subtypes can describe arbitrary two-dimensional shapes. See Figure A-2 and Figure A-10. A FEATURE is used to describe the location of a large ORGANIZATION. A small ORGANIZATION (down to an individual) may be associated more directly with a LOCATION via an ORGANIZATION-LOCATION. See Figure A-2.
Intel	SENSOR-TYPE[3]	A MILITARY-UNIT has associated MATERIEL-ITEM entities; the association records the number of such entities. SENSOR-TYPE is a subtype of MATERIEL-ITEM. See Figure A-4.
Communications	TELECOMMUNICATIONS-NETWORK-ELEMENT and its subtypes	A MILITARY-UNIT has associated MATERIEL-ITEM entities. A subtype of MATERIEL-ITEM is TELECOMMUNICATIONS-NETWORK-DEVICE, which has a many to many relationship with TELECOMMUNICATIONS-NETWORK-ELEMENT. Subtypes of TELECOMMUNICATIONS-NETWORK-ELEMENT describe various kinds of telecommunications devices. See Figure A-6.
SystemGroup	MATERIEL-ITEM and its subtypes, e.g. SENSOR-TYPE	A system is a type of materiel.[4] A MILITARY-UNIT has associated MATERIEL-ITEM entities, which describe materiel types. The association (via a HOLDING entity) records the number of such entities. SENSOR-TYPE is a subtype of MATERIEL-ITEM. See Figure A-4.

[2] For brevity, the subtype relationship between MILITARY-UNIT and ORGANIZATION will be omitted in the future.

[3] There is a proposal to add an entity named INFORMATION-REFERENCE to the AICDM. INFORMATION-REFERENCE is a pointer to any kind of relevant data. This entity might be relevant to intelligence.

[4] The OMSC methods for a SystemGroup return only properties related to system type and number. Future extensions to OMSC may require including AICDM entities that model materiel, not just materiel type.

OMSC Class	Related AICDM Entity	Relation to MILITARY-UNIT
Platform (ground, water, and space platforms)	MATERIEL	A platform can be a type of materiel. Since a Platform is an instance, a system may choose to associate a MATERIEL platform with a MILITARY-UNIT through MATERIEL-ORGANIZATION. See Figure A-7.
Platform (person platforms)	PERSON	A platform can also be a person. A MILITARY-UNIT has associations with a PERSON. See Figure A-4.
Platform (air platforms)	AIRCRAFT	None; currently, air platforms are represented separately from other types of materiel in the AICDM. However, there are no associations between a MILITARY-UNIT and INDIVIDUAL AIRCRAFT.
Platform (static platforms)	FACILITY	A platform can also be a facility (such as a missile silo). A MILITARY-UNIT has associations with FACILITY and FACILITY-TYPE. See Figure A-5.
PlatformInfo (ground, water, and space platforms)	• EQUIPMENT-TYPE • VEHICLE-TYPE	PlatformInfo gives signature information about a platform. If the platform is materiel, a MILITARY-UNIT has associated MATERIEL and MATERIEL-ITEM entities (see Figure A-4). The signature information for materiel that the AICDM can model includes dimensions, cross-sectional properties, and weight. The EQUIPMENT-TYPE and VEHICLE-TYPE entities model dimensions and weight. If a system models types of platforms, Figure A-4 shows how the EQUIPMENT-TYPE and VEHICLE-TYPE entities relate to a MILITARY-UNIT. If the system models individual platforms, Figure A-7 shows how MATERIEL is associated with a MILITARY-UNIT, and how MATERIEL is associated with MATERIEL-ITEM subtypes (for signatures that apply to a class of equipment).
PlatformInfo (person platforms)	PERSON	PlatformInfo gives signature information about a platform. If a platform is a person, a MILITARY-UNIT has associated PERSON entities (see Figure A-4) that can model certain physical characteristics of a person.
PlatformInfo (air platforms)	AIRCRAFT-TYPE	While individual AIRCRAFT are not associated with a MILITARY-UNITunit in the AICDM, the MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE can associate an AIRCRAFT TYPE with an ORGANIZATION.
PlatformInfo (static platforms)	None.	The AICDM does not contain relevant information for facilities.

OMSC Class	Related AICDM Entity	Relation to MILITARY-UNIT
Attrition	• ORGANIZATION-TYPE-ORGANIZATION-HOLDING-ESTIMATE • PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE • PERSON • MATERIEL	A MILITARY-UNIT has associations with materiel and persons, and with types of materiel and persons. In the former case, the number of relationships record the number of associated entities (see Figure A-4 and Figure A-7). In the latter case, the associations record the number of associated entities (see Figure A-4).
Logistics	• ORGANIZATION-TYPE-ORGANIZATION-HOLDING-ESTIMATE • PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE	A MILITARY-UNIT has associations with materiel and persons, and with types of materiel and persons. In the former case, the number of relationships record the number of associated entities (see Figure A-4 and Figure A-7). In the latter case, the associations record the number of associated entities (see Figure A-4).
Maintenance	• PERSON • PERSON-OPERATIONAL-STATUS • PERSON-CAPABILITY-ESTIMATE • MATERIEL • MATERIEL-OPERATIONAL-STATUS • MATERIEL-CAPABILITY-ESTIMATE	A military-unit has associations with PERSON and MATERIEL entities. These entities in turn have associations with operational status entities. See Figure A-4 and Figure A-7. The specification of the actual functionality, such as maintenance and supply may be specified in other portions of the DoD Data Architecture, which at present do not form part of the AICDM. But in the first instance it should be possible to assign these types of high level capabilities to any given military unit in charge of performing these tasks.
Supply	• PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE • PERSON • MATERIEL • ORGANIZATION-TYPE-CAPABILITY-NORM • ORGANIZATION-CAPABILITY-ESTIMATE	If a system models quantities, a MILITARY-UNIT's associations with the entities that describe types of persons and materiel include the quantities of each type. See Figure A-4. If a system models individual entities, a MILITARY-UNIT has associations with individual entities. The number of extant relationships determine the unit's supply characteristics. See Figure A-4. A MILITARY-UNIT has associated ORGANIZATION-CAPABILITY-ESTIMATE entities that can model maximum supply capabilities. The ORGANIZATION-TYPE-CAPABILITY-NORM entity can model certain capabilities for classes of military units. See Figure A-3.

OMSC Class	Related AICDM Entity	Relation to MILITARY-UNIT
C2	• ACTION, including subtypes • PLAN • MISSION • TASK	A Military-Unit has associations with action, plan, mission, and task entities. See Figure A-8.

The degree of Conceptual level alignment of the Unit standard object is the average of the degrees of alignment of its entities (56%).

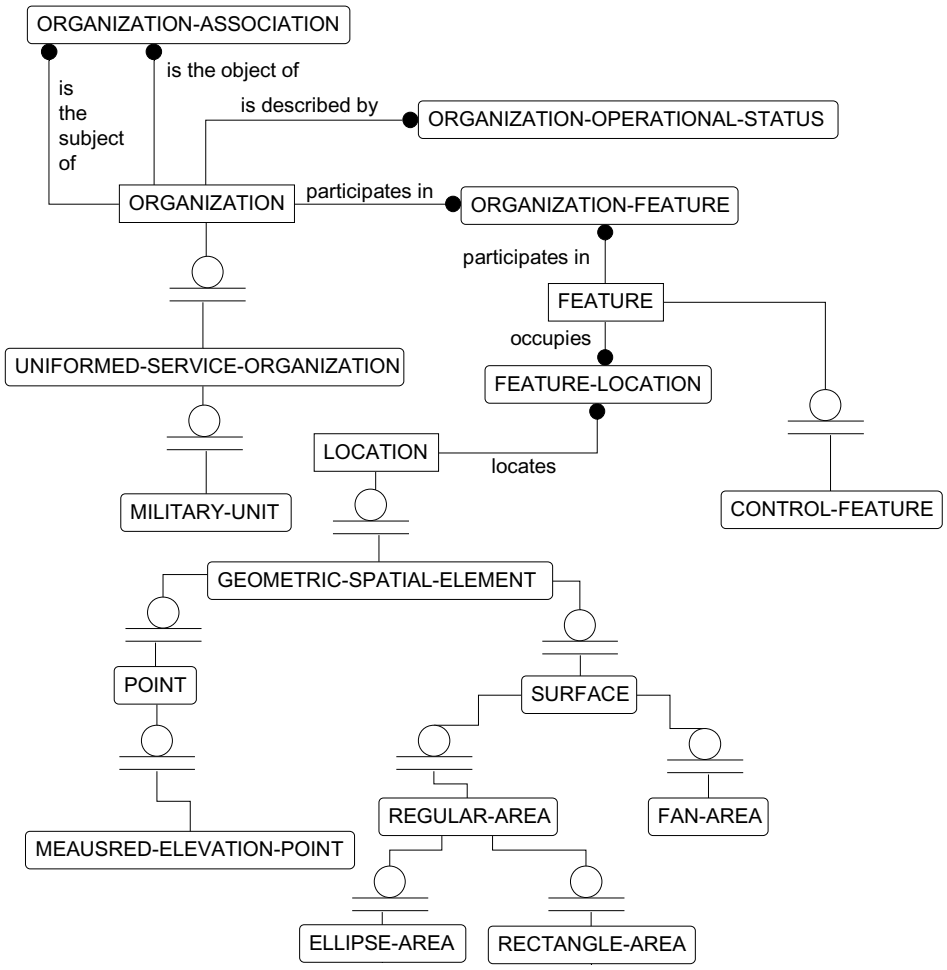


Figure A-2. AICDM Organizations and Locations

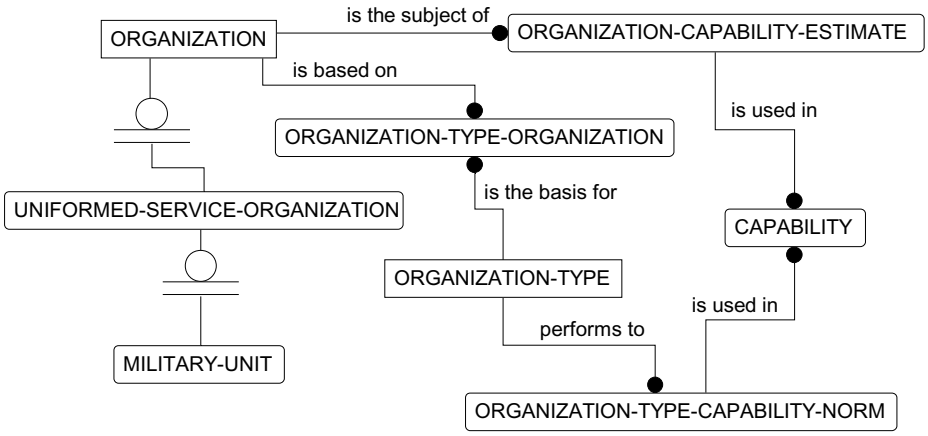


Figure A-3. AICDM Capabilities

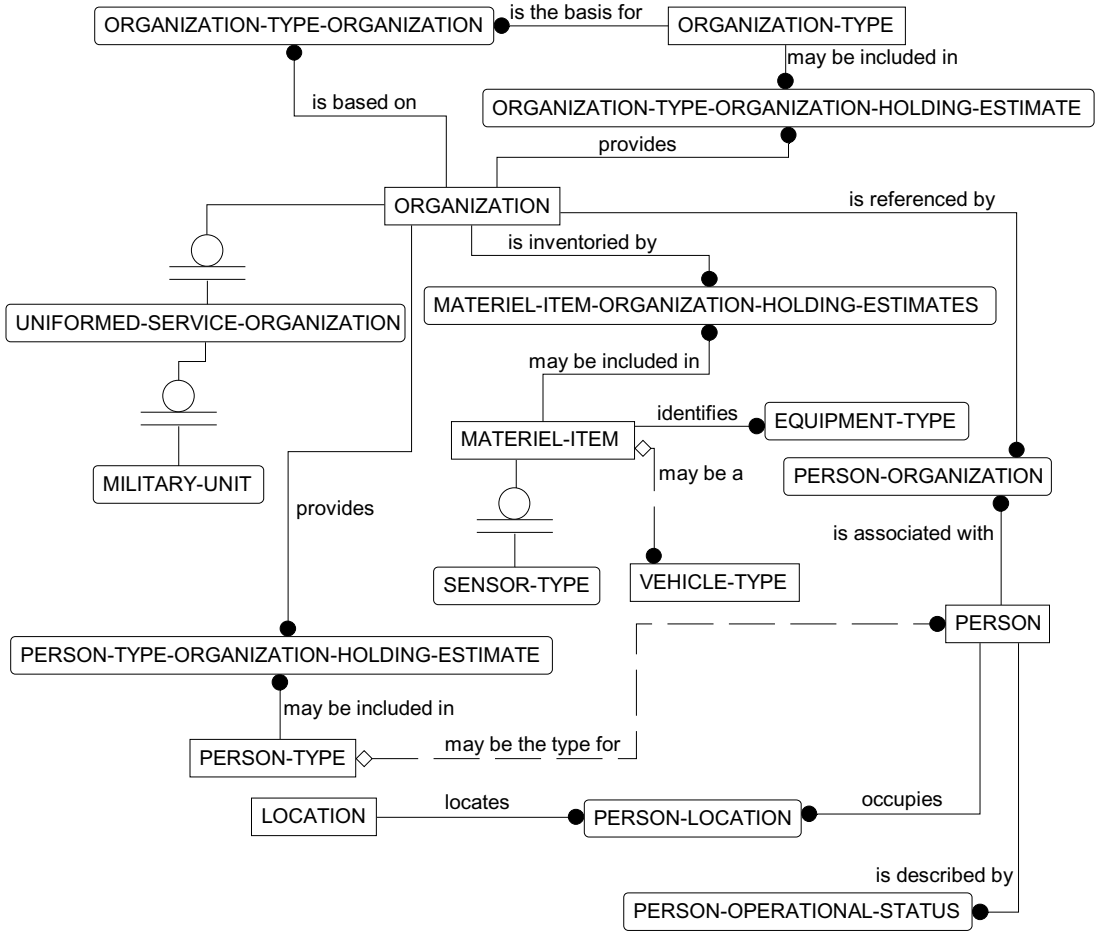


Figure A-4. AICDM Materiel and Personnel Quantities

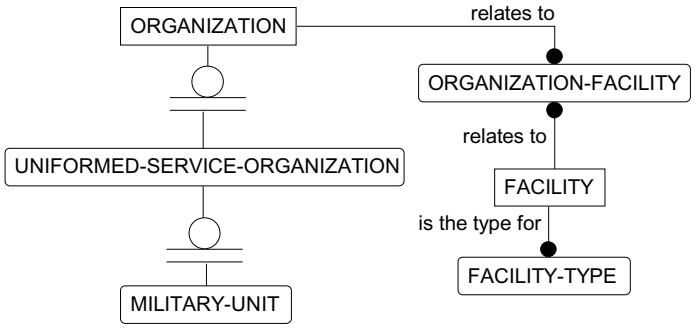


Figure A-5. AICDM Organization and Facilities

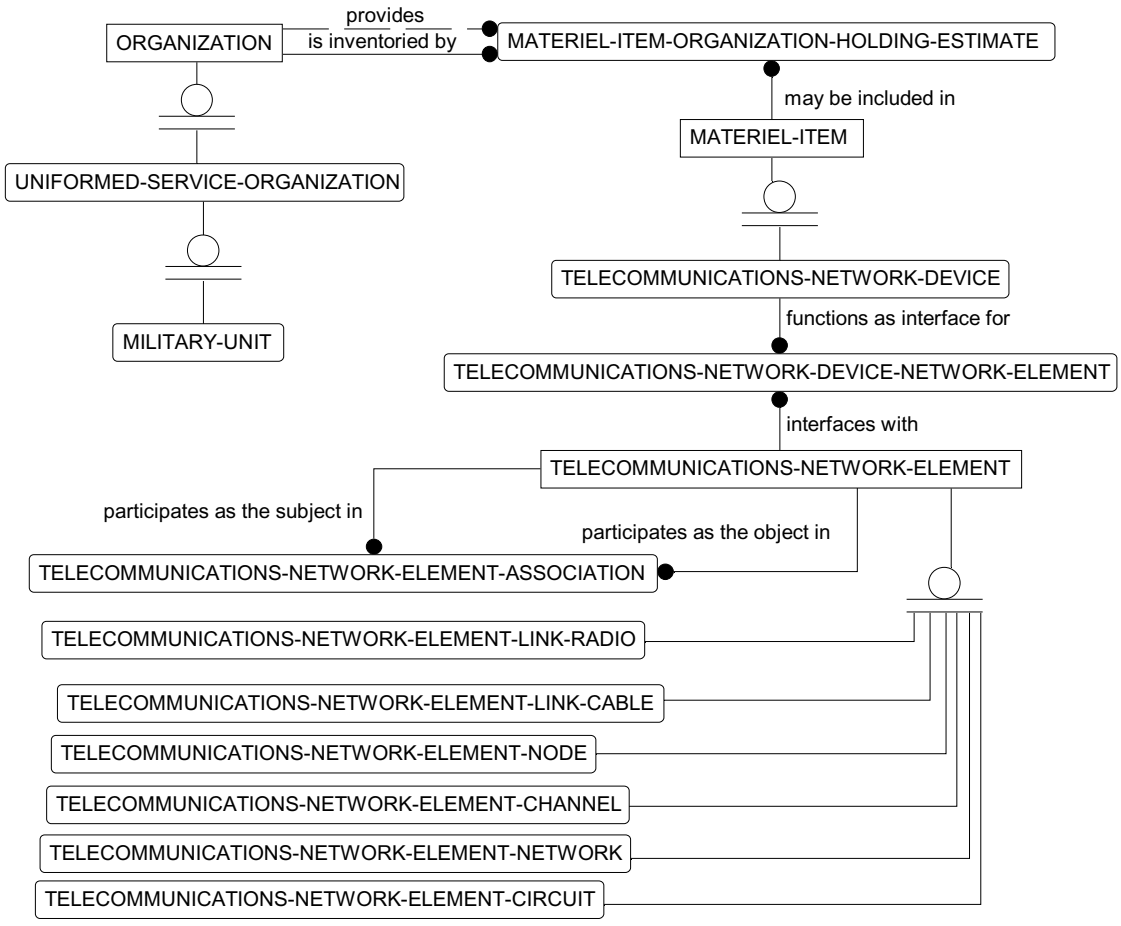


Figure A-6. AICDM Telecommunications

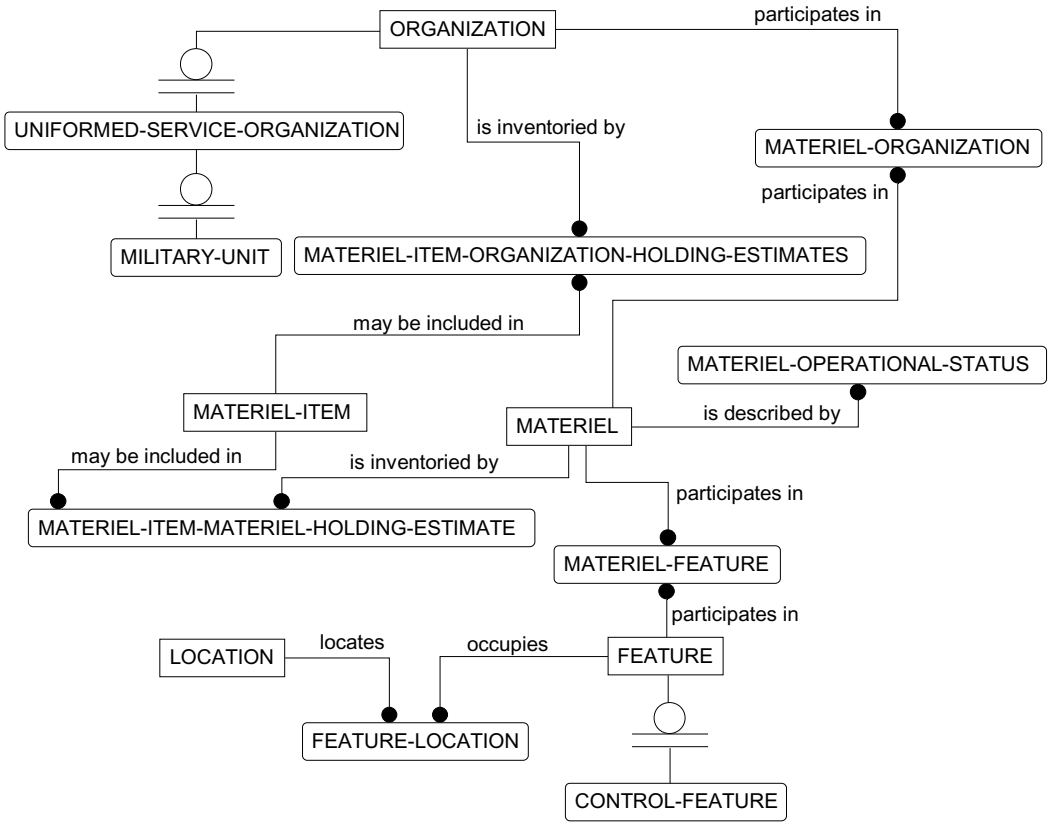


Figure A-7. AICDM Materiel

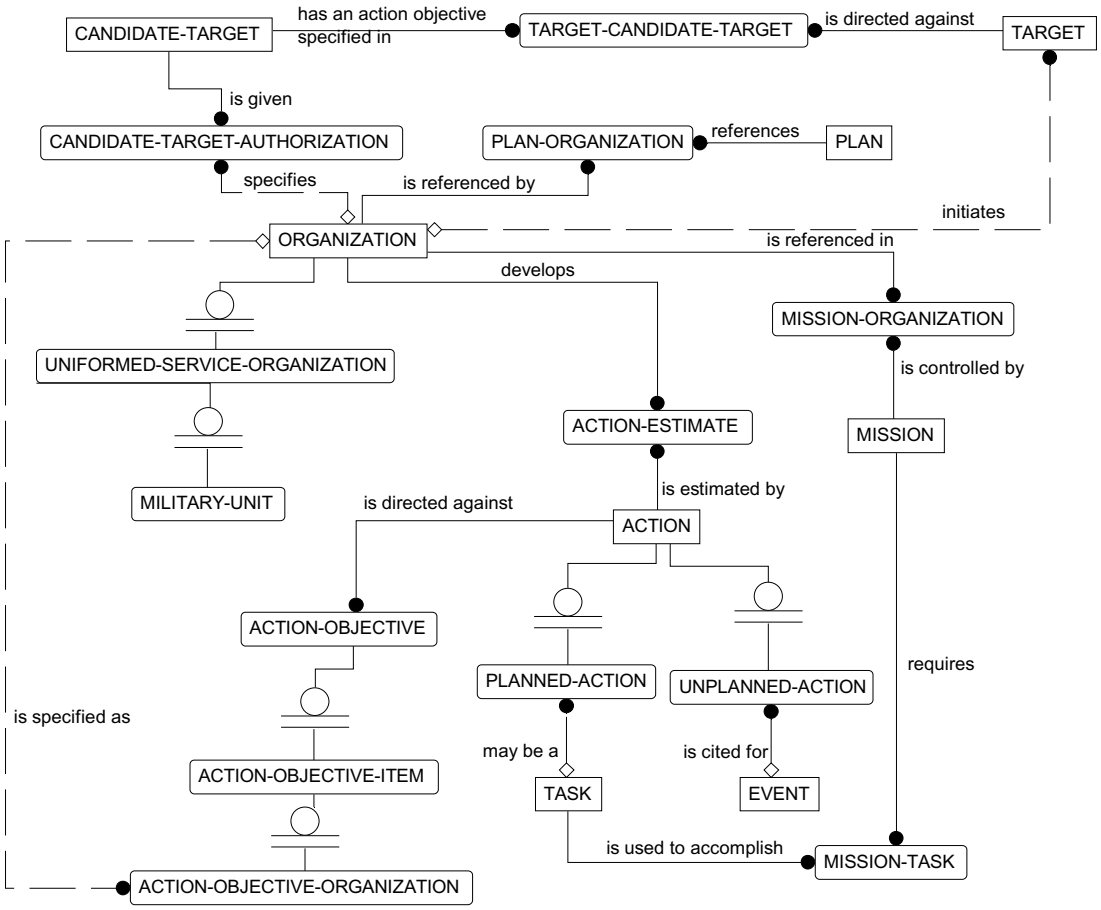


Figure A-8. AICDM Actions, Missions, Plans, and Tasks

A.1.1. Unit Class (Entity Level)

The Unit class aligns with the AICDM entity MILITARY-UNIT, the focal entity of the Military Unit view. The names of Unit's methods show the need for other AICDM entities and relationships if an instance of Unit is to be fully modeled in the AICDM, because the attributes of MILITARY-UNIT do not appear to encompass the range of concepts those methods comprise (see Figure A-9). These entities are shown in Table A-3.

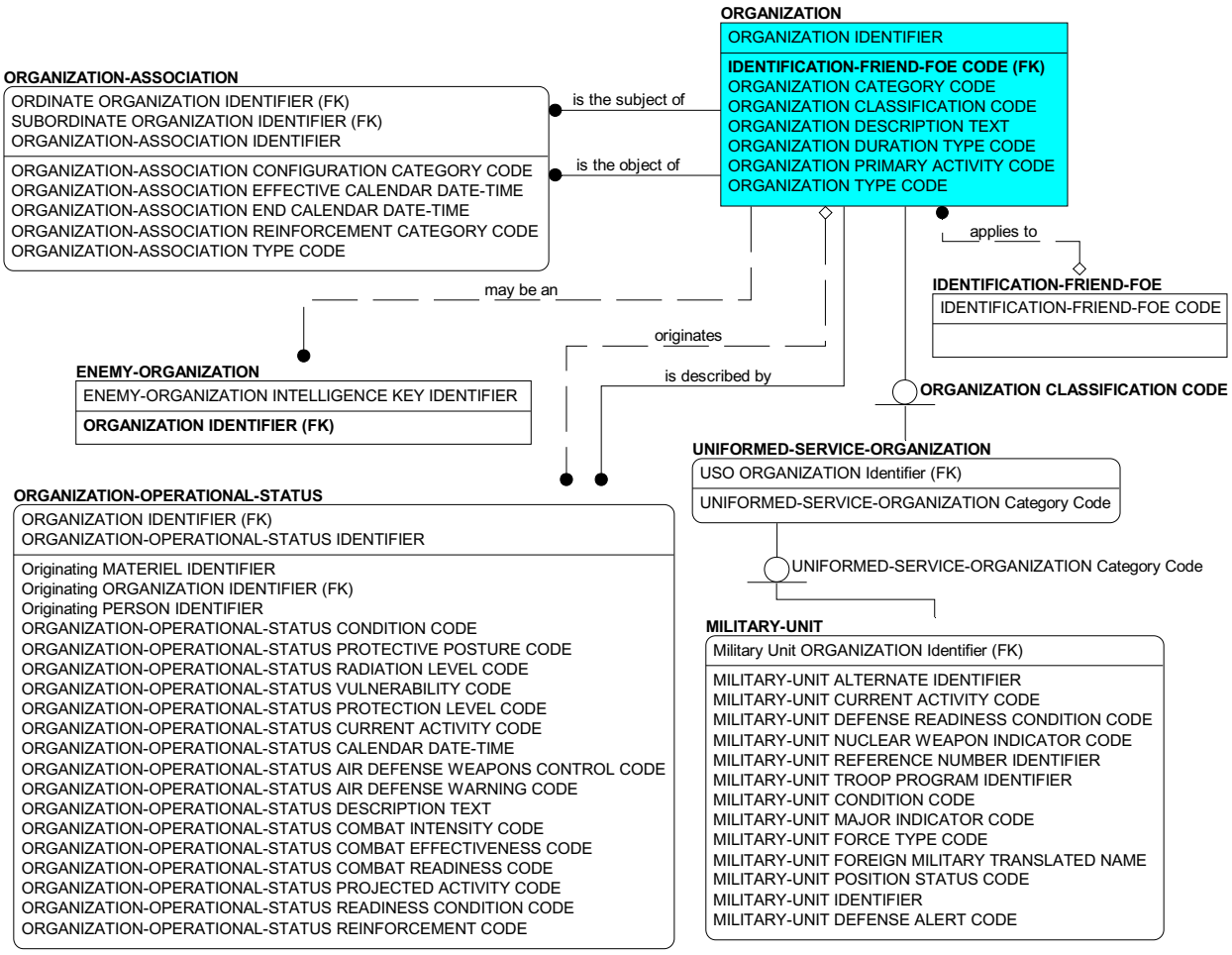


Figure A-9. AICDM Relationship between MILITARY-UNIT and ORGANIZATION

Table A-3. AICDM entities that align to the Unit class at the State level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• POINT • MEASURED-ELEVATION-POINT	getLocation()	A MILITARY-UNIT is associated with a CONTROL-FEATURE, which has an associated LOCATION. GEOMETRIC-SPATIAL-ELEMENT is a subtype of LOCATION. POINT is a subtype of GEOMETRIC-SPATIAL-ELEMENT. MEASURED-ELEVATION-POINT is a subtype of POINT. See Figure A-2.
None.	getVelocity()	N/A
• ORGANIZATION • MILITARY-UNIT	getID()	MILITARY-UNIT is a subtype of ORGANIZATION.
• ORGANIZATION • ORGANIZATION-ASSOCIATION	getSide() ORGANIZATION is also necessary as a link to other entities	An ORGANIZATION has a hierarchical structure defined by relationships with the ORGANIZATION-ASSOCIATION entity. See Figure A-9.

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
ORGANIZATION-OPERATIONAL-STATUS	• getPosture() • getStatus()	A MILITARY-UNIT is described by an ORGANIZATION OPERATIONAL STATUS. See Figure A-9.
TASK	getMission()	A MILITARY-UNIT has a MISSION, which is composed of one or more TASKs. See Figure A-8.
MILITARY-UNIT	getEchelon()	Identical (the same entity)
• POINT • MEASURED-ELEVATION-POINT	move()	A MILITARY-UNIT is associated with a CONTROL-FEATURE, which has an associated LOCATION. A subtype of LOCATION is POINT (for two dimensions); a subtype of POINT is MEASURED-POINT (for three dimensions). See Figure A-2.
• ORGANIZATION-TYPE-ORGANIZATION HOLDING ESTIMATE • PERSON-TYPE-PERSON HOLDING ESTIMATE	determineAttrition()	A MILITARY-UNIT has associated PERSON-TYPE and MATERIEL-ITEM entities. The association records the number of entities of the type. See Figure A-4 and Figure A-7.

The degree of Entity level alignment of the Unit class is the average of the degrees of alignment of its methods (51%).

A.1.1.1. The getLocation() Method (State Level)

The OMSC defines the result of the getLocation() method as "typically … center of mass or some other point location…." [HB 1998A] The following ambiguities in this description limit the alignment analysis:

- Apparently, location can be something other than a point. It might, for example, be an arbitrary geometrical shape encompassing an entire unit, rather than the unit's center of mass.

  We ignore the cases where location is not a point, and align getLocation() to those AICDM entities that model points. The rationale for this decision is that the OMSC has a method UnitGeometry.getShape(), which returns a unit's shape. To include geometrical shapes as possible values of getLocation() would be redundant.

- The OMSC does not state how center of mass is computed. The implication of this ambiguity is that a unit's location cannot be calculated based on knowledge of its shape.

The OMSC has a standard object Location. The result of the getLocation() method aligns to the AICDM at the State level to the same degree that Location aligns to the AICDM (37%). See Section A.3. Figure A-10 shows the AICDM structures for specifying the location of military units.

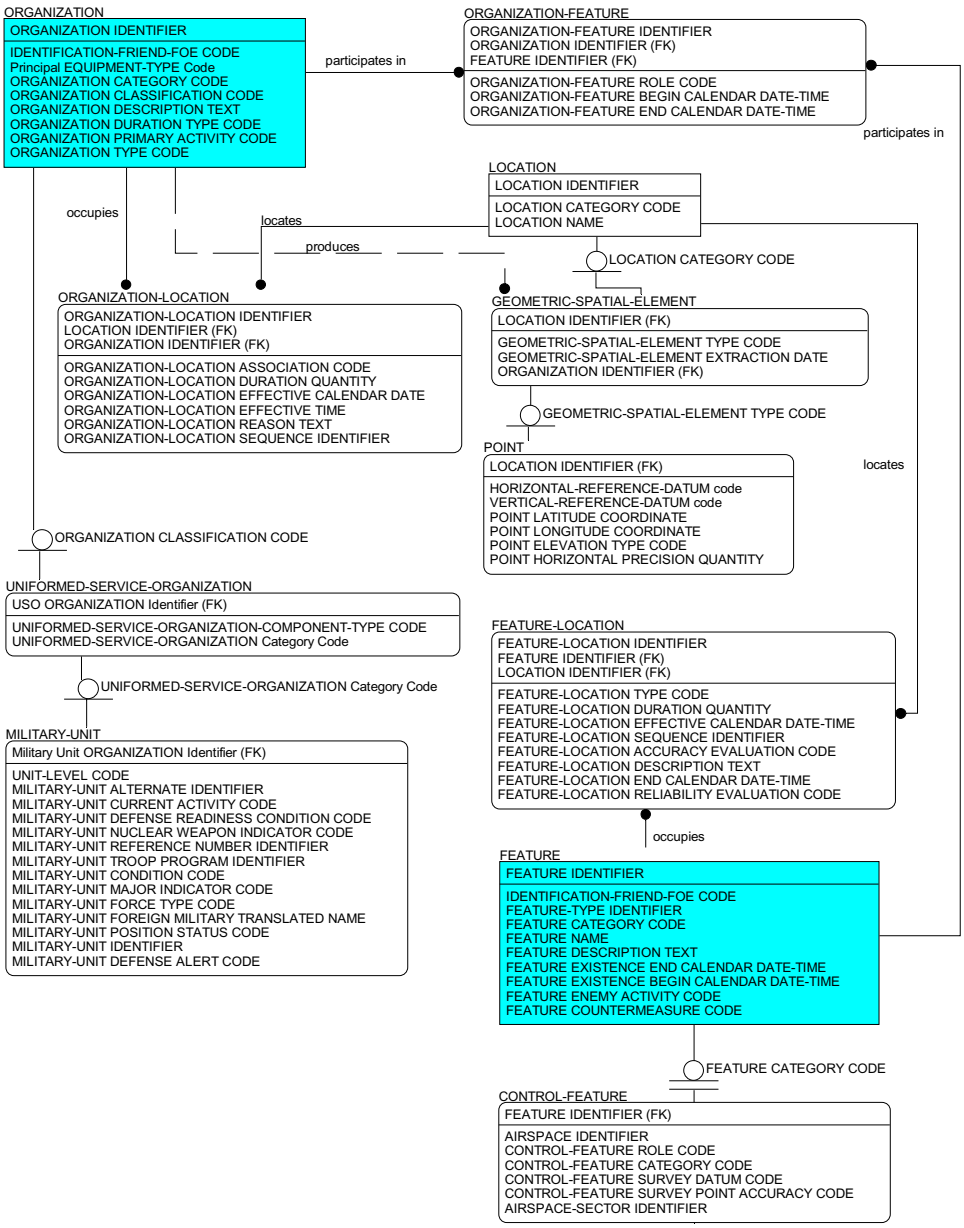


Figure A-10. AICDM Structures for Locating Military Units

A.1.1.2. The getVelocity() Method (State Level)

This method's description indicates that it returns the velocity of a Unit. The AICDM does not model motion of organizations, nor does it model orientation. (REGULAR-AREA and FAN-AREA have an ORIENTATION ANGLE attribute, but the attribute refers to the angle of

the geometric figure, not the direction component of the figure's velocity.) There is **no alignment at the State level** (0%) between getVelocity() and the AICDM.[5]

A.1.1.3. The getID() Method (State Level)

The following ambiguities in the description of the getID() method limit alignment analysis:

- There is no indication of standards for assigning IDs to units. Standards might come from such considerations as the following:
  - How will the ID be used? For example, if it will be displayed on a screen, it must be fairly short.
  - Should it relate to real unit identifiers?

  The alignment analysis assumes only that IDs must be unique. AICDM IDs must be IDs of real units; OMSC's need not. To achieve perfect alignment might require some automated translation system that maps between the schemes.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC getID() method. The AICDM has many attributes that can represent an ID (see Figure A-9). The getID() method might map to any of the following AICDM attributes of MILITARY-UNIT:

- MILITARY-UNIT ALTERNATE IDENTIFIER
- MILITARY-UNIT REFERENCE NUMBER ID
- MILITARY-UNIT TROOP PROGRAM IDENTIFIER
- MILITARY-UNIT FOREIGN MILITARY TRANSLATED NAME
- MILITARY-UNIT IDENTIFIER

The getID() method might also map to the ORGANIZATION-IDENTIFIER attribute of ORGANIZATION.

The AICDM and the OMSC do not align perfectly with respect to getID() because it is not clear which of these attributes models the type of ID returned by getID(). This cannot be determined without more detail from the OMSC.

---

[5] There is a proposal to add velocity and orientation attributes to the pertinent entities in the AICDM. These additions will support the OMSC requirements associated with the getVelocity() method.

A.1.1.4. The getSide() Method (State Level)

The following ambiguities in the description of the getSide() method limit alignment analysis:

- There is no indication of the maximum number of sides that must be supported. At one time it was assumed that the only sides would be "blue" and "red", but with the advent of coalition warfare the number of possible sides has grown. Note that WARSIM has a contractual requirement to support at least 36 different sides.

  The alignment analysis does not assume a limit on the number of sides that must be modeled.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getSide() method. The alignment is not perfect because the built-in attributes of the AICDM express a more limited model of sides than that of the OMSC:

- An ORGANIZATION has an IDENTIFICATION-FRIEND-FOE CODE attribute (a foreign key).[6] This is more limited than the concept of arbitrary coalitions and alignments required by the OMSC.

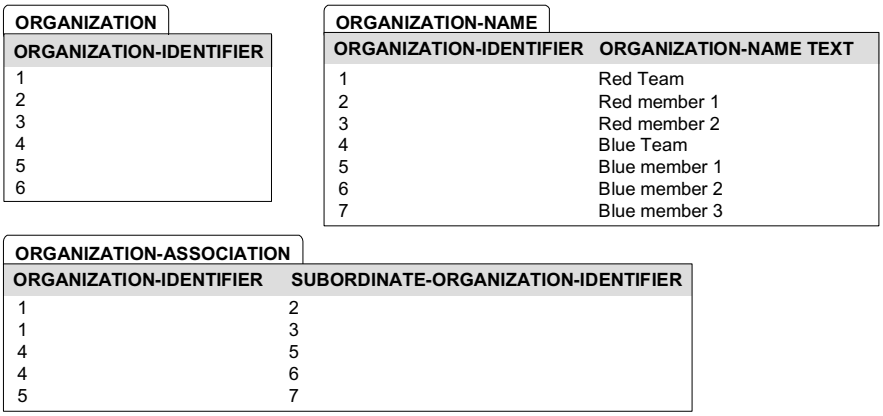
- An ORGANIZATION "pertains to" one or more countries, and an ENEMY-ORGANIZATION is a subtype of ORGANIZATION (see Figure A-9). However, this relationship cannot model more than two sides (friend and foe).

The alignment analysis uses the following approach to model a faction (or coalition). Each faction is modeled as an ORGANIZATION. The name of this ORGANIZATION entity (given in the ORGANIZATION-NAME entity) is the name of the faction. The constituents of the faction are associated with the side through the ORGANIZATION-ASSOCIATION entity, which can form a hierarchical relationship of organizations. See Figure A-2.

Figure A-11 shows an example of entity instances populating AICDM tables. These instances represent two factions: "Red team" and "Blue team". Red team consists of two member, Red member 1 and Red member 2. Blue team consists of 3 members, Blue members 1 through 3.

---

[6] Previous versions of the AICDM modeled this as an attribute of each of the battlefield objects. However, in order to make re-use of the attributes more readily these attributes have been externalized as an independent entity. The current attributes will be replaced by a non-identifying relationship from this entity to the ones requiring those types of values.



ORGANIZATION

ORGANIZATION-IDENTIFIER
1
2
3
4
5
6

ORGANIZATION-NAME

ORGANIZATION-IDENTIFIER	ORGANIZATION-NAME TEXT
1	Red Team
2	Red member 1
3	Red member 2
4	Blue Team
5	Blue member 1
6	Blue member 2
7	Blue member 3

ORGANIZATION-ASSOCIATION

ORGANIZATION-IDENTIFIER	SUBORDINATE-ORGANIZATION-IDENTIFIER
1	2
1	3
4	5
4	6
5	7

Figure A-11. Example of Faction Representation

For brevity, Figure A-11 omits several attribute values. It shows only how organizations can be arranged to form a coalition. There is still no specification of enmity, or lack thereof, between the two coalitions.

A problem with using this convention for modeling sides is that the ORGANIZATION TYPE CODE attribute has no value that denotes a coalition or faction. Neither, for that matter, does the ORGANIZATION-ASSOCIATION TYPE CODE attribute. Possible values include HAS FULL COMMAND OVER, REPORTS TO, CONTROLS, and IS IN DIRECT SUPPORT OF, depending on the type of coalition. VALUE

A.1.1.5. The getPosture() Method (State Level)

The following ambiguities in the description of the getPosture() method limit alignment analysis:

- Possible postures are described using a few examples. The list cannot be regarded as complete.

  The alignment analysis attempts to use the doctrinal interpretation of the term.

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the getPosture() method. This statement is based on the observation that few AICDM attributes model posture as the term is generally used in M&S systems; and those attributes that model posture do so to a limited degree. The following is a list of candidate AICDM attributes (see Figure A-9):

- ORGANIZATION-OPERATIONAL-STATUS PROTECTIVE POSTURE CODE
- ORGANIZATION-OPERATIONAL-STATUS CURRENT ACTIVITY CODE
- MILITARY-UNIT CURRENT ACTIVITY CODE

However, none of these names seems adequate to accommodate the full range of postures suggested by the OMSC's description of the method.

An examination of the domains of the above attributes confirms the previous paragraph. Each can model some types of postures. Not even the combined set can model all. VALUE

A.1.1.6. The getStatus() Method (State Level)

The following ambiguities in the description of the getStatus() method limit alignment analysis:

- Possible status are described using a few examples. The list cannot be regarded as complete.

  The alignment analysis attempts to use doctrinal interpretation of the term.

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the getStatus() method. This statement is based on the observation that few AICDM attributes model status as the term is generally used in M&S systems; and those attributes that model status do so to a limited degree. In the AICDM the following attributes are possibilities, but as with getPosture() none of the names seems adequate to accommodate the full range of statuses suggested by the OMSC's description of the method (see Figure A-9):

- ORGANIZATION-OPERATIONAL-STATUS COMBAT EFFECTIVENESS CODE
- ORGANIZATION-OPERATIONAL-STATUS COMBAT READINESS CODE
- MILITARY-UNIT.MILITARY-UNIT CURRENT READINESS CONDITION CODE
- MILITARY-UNIT DEFENSE READINESS CONDITION CODE

A problem with aligning these attributes to the result of the getStatus() method is that the attribute domains are enumerated code values. By contrast, the results of getStatus() can be a numeric value (e.g., a percent effectiveness). VALUE

Furthermore, an examination of the domains of the above attributes confirms that the attributes cannot model a wide range of statuses. VALUE

A.1.1.7. The getMission() Method (State Level)

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the getMission() method (see Figure A-12). In M&S systems, an OMSC mission is prose. It therefore aligns with the TASK-DESCRIPTION-TEXT attribute of an AICDM TASK entity. However, the alignment is unidirectional. An AICDM TASK may be a hierarchy. This hierarchy might map to an AICDM mission using some standard algorithm for amalgamating the task descriptions in the hierarchy, but the original structure

couldn't necessarily be recovered from getMission(). In other words, the AICDM can model the OMSC, but not vice versa.

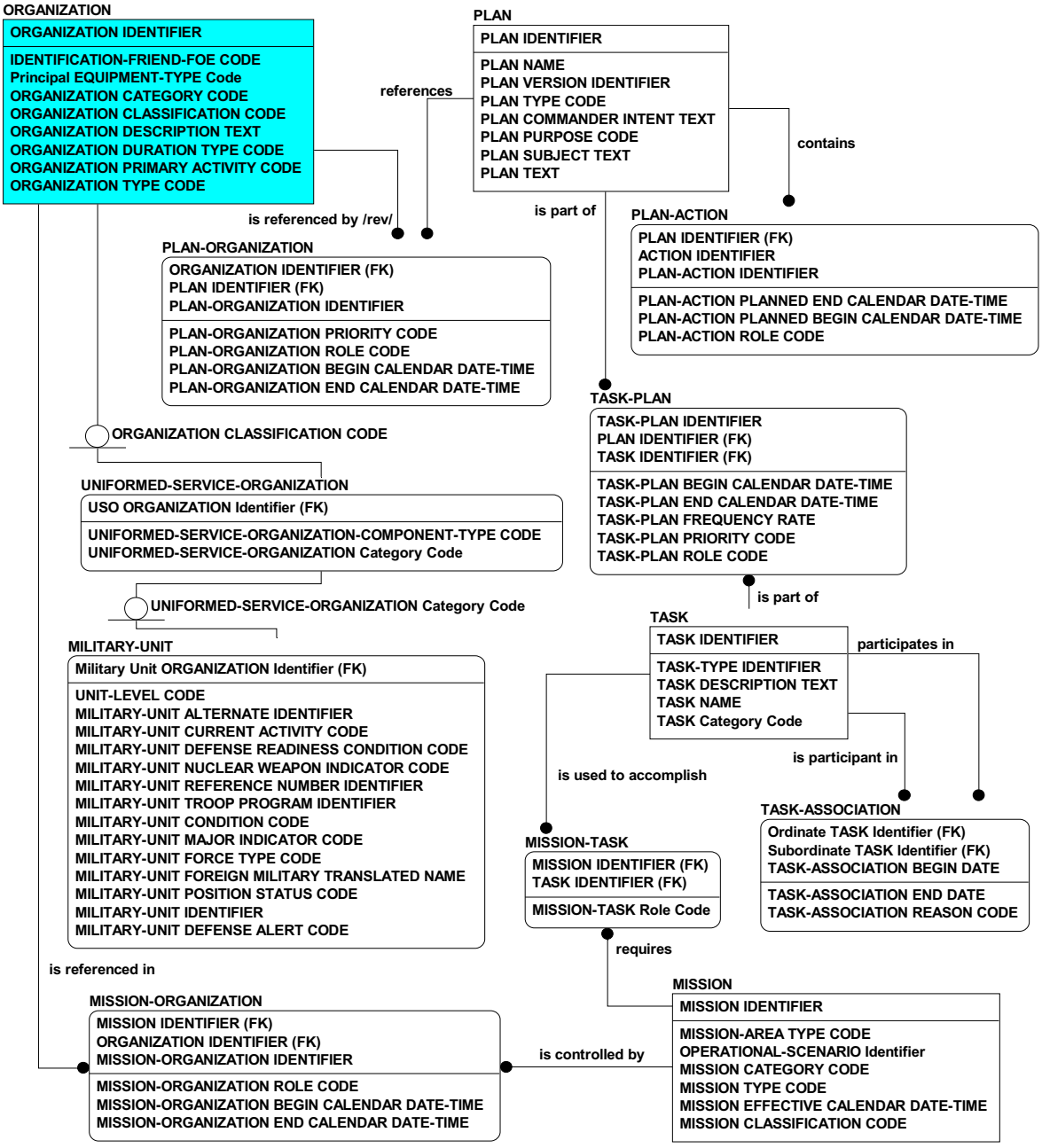


Figure A-12. AICDM Structures for Missions and Tasks

A.1.1.8. The getEchelon() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getEchelon() method. The return value of getEchelon() maps to the MILITARY-

UNIT UNIT-LEVEL CODE attribute of the MILITARY-UNIT entity (see Figure A-9). The values for this attribute cover all the standard terms for echelons. (We assume that M&S systems use standard terms as well.)

A.1.1.9. The move() Method (State Level)

The following ambiguities in the description of the move() method limit alignment analysis:

- The specification does not state the parameters of the method. How is the next location determined? Without this knowledge, alignment analysis cannot determine if all reasonable variations of movement can be modeled in the AICDM. An example of one that could not be modeled is "move forward", because the AICDM does not model unit orientation.

- The OMSC states that move() "advances a unit towards its next location". This seems to imply that the next location is known in advance. If so, parameters might not be needed. But the OMSC has no method to specify a next location.

- The OMSC does not specify a model for changes to a unit's supplies and readiness as caused by movement. Movement can drain fuel, reduce troop readiness, etc. It is impossible to determine if the AICDM contains all the attributes necessary to model the effects of movement in an M&S system.

This is a behavioral method. The following are some possible state changes:

- A change to a unit's location. The AICDM is able to model this change, because an ORGANIZATION has (via a CONTROL-FEATURE) a LOCATION (see Figure A-10). The association is through relationships involving an entity ORGANIZATION-FEATURE, which has attributes ORGANIZATION-FEATURE BEGIN CALENDAR DATE-TIME and ORGANIZATION-FEATURE END CALENDAR DATE-TIME. These attributes are particularly useful in keeping a history of state, if desired.

- A change to a unit's supplies. The HOLDING entities, or associations to MATERIEL entities, can model changes in these quantities. See Figure A-13.

- A change to the condition of a unit's supplies and personnel (e.g., equipment breakage, personnel injuries). If the M&S system records individual entities, the AICDM can model these changes. If the M&S system models only quantities, the AICDM cannot model these changes.

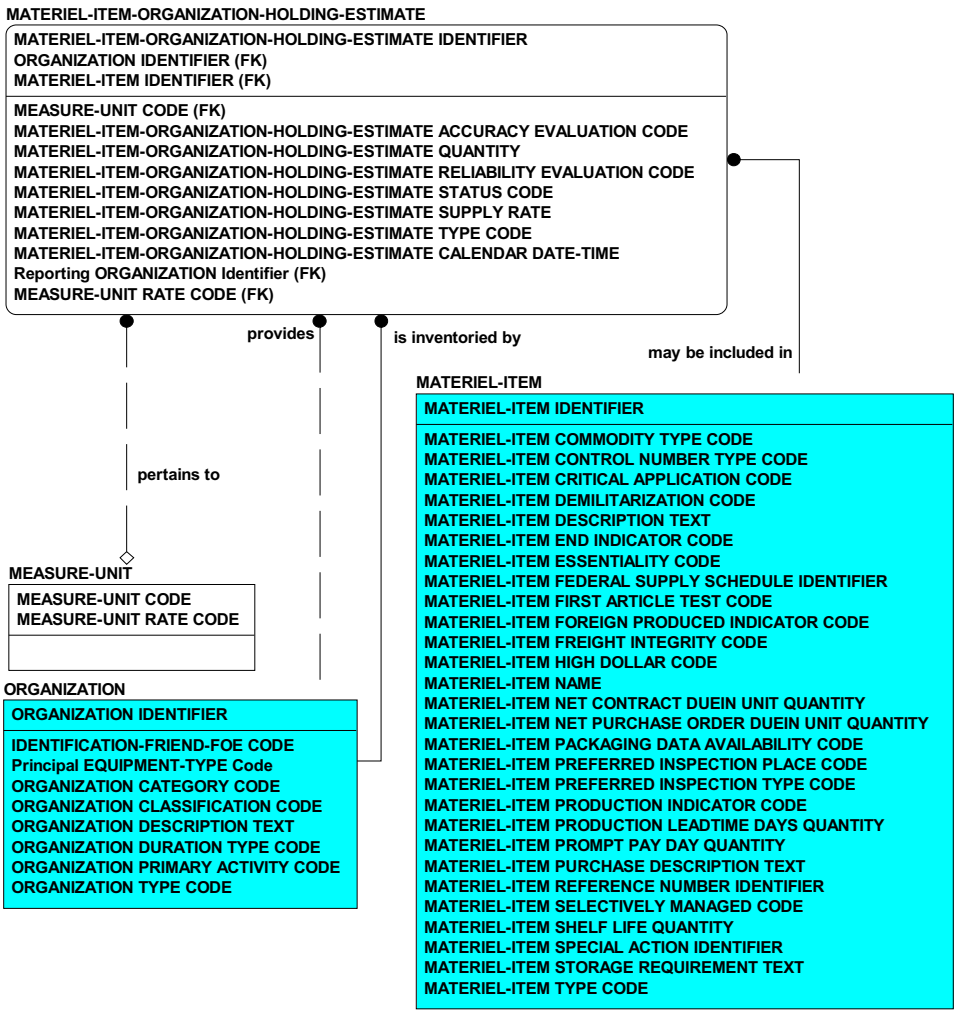


Figure A-13. AICDM Structures for Specifying Materiel Holdings

Of these state changes, the OMSC only mentions location as an effect of the move() method.

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the move() method. Assuming that movement is specified based on either relative or absolute direction, it seems reasonable to conclude that the AICDM can model half of all movement operations, because the AICDM can model absolute but not relative direction.

A.1.1.10. The determineAttrition() Method (State Level)

The following ambiguities in the description of the determineAttrition() method limit alignment analysis:

- The OMSC does not specify units for attrition (percentage? absolute values? specific entities removed?)

- An analysis of attrition must occur with respect to some previous state. The OMSC does not describe how that state is specified. It would have to `be either fixed or specified as a parameter.

- The description does not say whether invoking the method actually causes the attrition or just calculates a value that connotes attrition. As there is a separate causeAttrition() method, determineAttrition() probably only calculates a value.

The determineAttrition() method aligns indirectly. Attrition is calculated relative to some previous state, based on predefined unit characteristics that are deemed to be attrition. These characteristics are described in Section . The determineAttrition() method aligns to the same degree as the causeAttrition() method (75%).

A.1.2. UnitGeometry Class (Entity Level)

The UnitGeometry class describes two characteristics of a unit: shape and orientation. Table A-4 shows the AICDM entities taken from the Conceptual level view of a unit that relate to geometry.

Table A-4. AICDM entities that align to the UnitGeometry class at the State level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
SURFACE and its subtypes	getShape()	A MILITARY-UNIT has (via a CONTROL-FEATURE) an associated LOCATION. GEOMETRIC-SPATIAL-ELEMENT is a subtype of LOCATION. SURFACE is a subtype of GEOMETRIC-SPATIAL-ELEMENT. See Figure A-2 and Figure A-14.

The degree of Entity level alignment of the UnitGeometry class is the average of the degrees of alignment of its methods (50%).

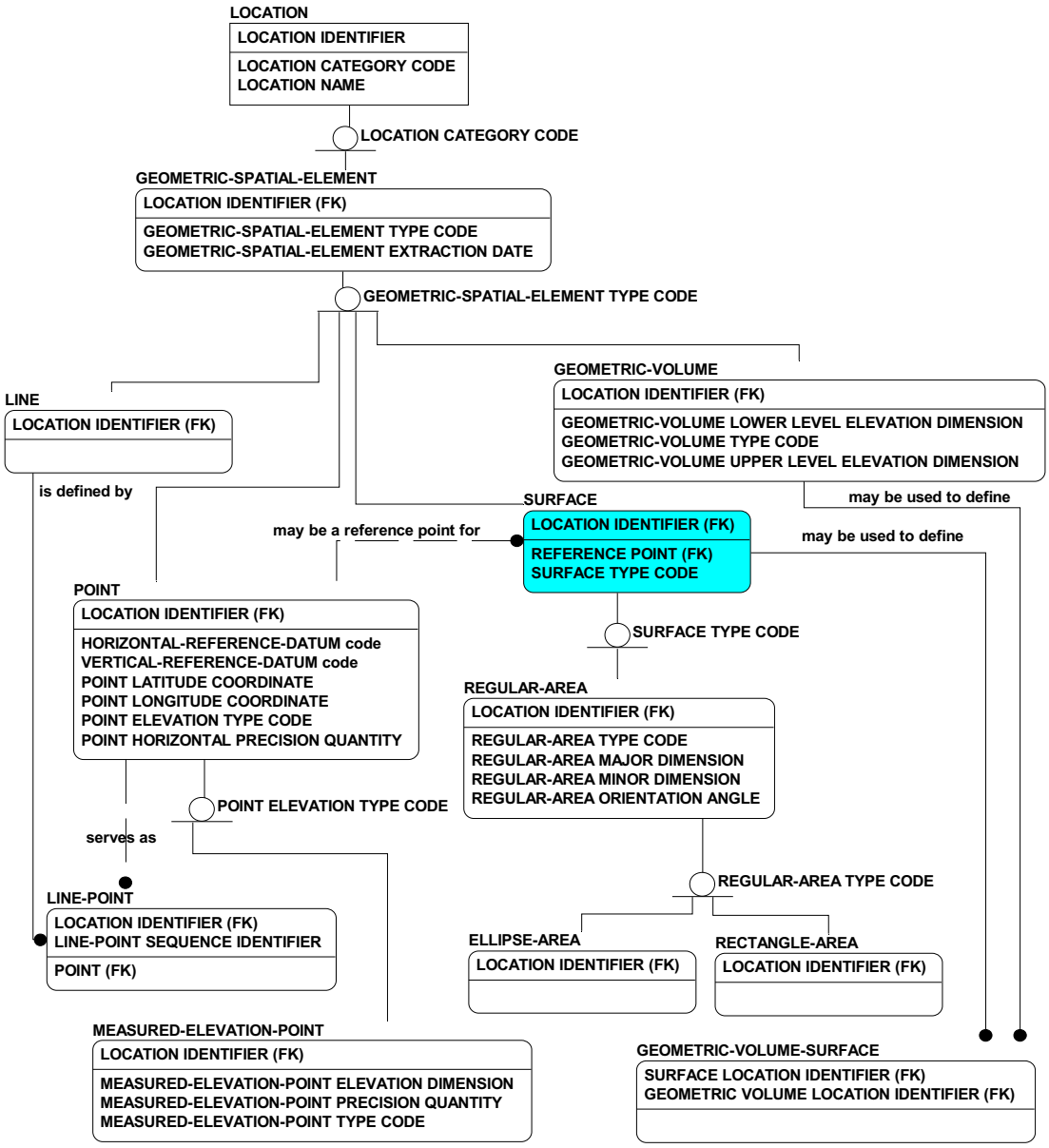


Figure A-14. AICDM Structures for Specifying Surfaces

A.1.2.1. The getShape() Method (State Level)

The following ambiguities in the description of the getShape() method limit alignment analysis:

- The OMSC does not indicate the nature or generality of bounding shapes.

  The AICDM has a rich set of possible shapes for a unit. It seems safe to assume that the AICDM can model any shape that getShape() would return. (the AICDM is richer

in two than in three dimensions, but there is no indication that getShape() will return any arcane three dimensional shapes.)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getShape() method. The return value of the getShape() method maps to a subtype of SURFACE.

A.1.2.2. The getOrientation() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getOrientation() method. The AICDM does not model unit orientation. The getOrientation() method does not align to any AICDM entities or attributes.

A.1.3. Intel Class (Entity Level)

Table A-5. AICDM entities that align to the Intel class at the State level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
SENSOR-TYPE	collect()	A MILITARY-UNIT has associated MATERIEL-ITEM entities; the association records the number of such entities. SENSOR-TYPE is a subtype of MATERIEL-ITEM. See Figure A-4.
• TARGET • CANDIDATE-TARGET • FEATURE	reportContacts()	A MILITARY-UNIT has associated feature entities. See Figure A-2. A MILITARY-UNIT also has associated TARGET and CANDIDATE-TARGET entities. See Figure A-15.

The degree of Entity level alignment of the Intel class is the average of the degrees of alignment of its methods (37%).

A.1.3.1. The collect() Method (State Level)

The following ambiguities in the description of the collect() method limit alignment analysis:

- The OMSC does not specify how the organic sensor assets of a unit are defined.
- Detection can involve more than turning on a sensor. The sensor might be directed against some feature, other organization, etc. The OMSC does not define how search capabilities may be effected other than turning them on.
- The OMSC has no method to terminate local detection.

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the collect() method. Nothing about how the collect() method aligns to the AICDM can be determined at the State level.

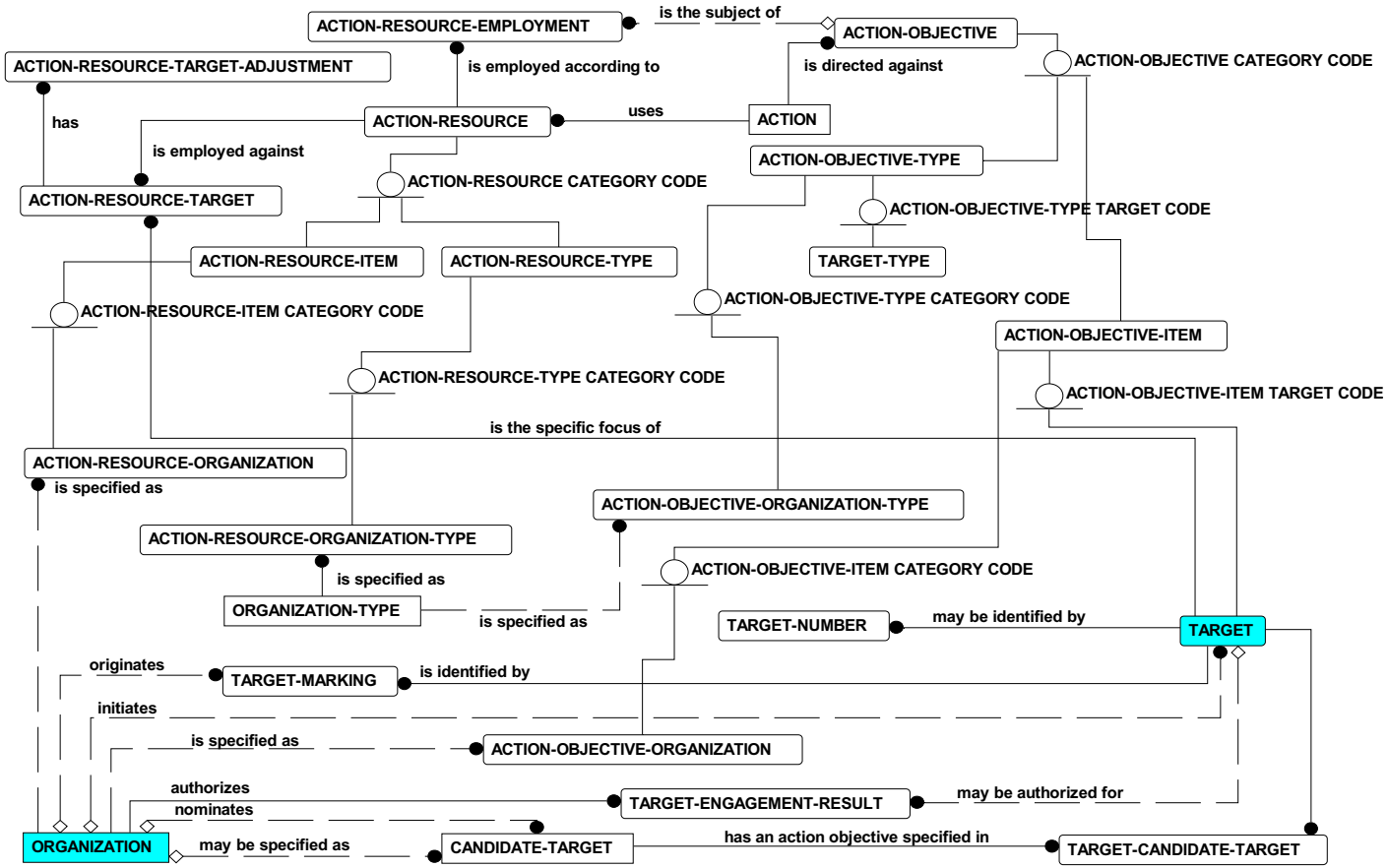


Figure A-15. AICDM Structures for Specifying Targets

A.1.3.2. The reportContacts() Method (State Level)

The following ambiguities in the description of the reportContacts() method limit alignment analysis:

- The OMSC does not specify the nature of results, nor how they might be used. The only other method associated with a Unit that might use intelligence is C2.doC2().

There is **a high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the reportContacts() method. The OMSC does not specify the nature of the method's results, but it is likely that a contact can be modeled either as a TARGET or a FEATURE. In particular, the AICDM defines a FEATURE as "of military significance." However, the ambiguities in the method's description preclude perfect alignment.

Currently, the AICDM lacks domain values to associate organizations as contacts. If additional domain values for the role codes in the associative entities related to ORGANIZATION are made part of future versions of the AICDM (e.g., "contacted by"), then ORGANIZATION-ASSOCIATION, ORGANIZATION-MATERIEL, ORGANIZATION-PERSON, etc., could be used to capture this OMSC requirement. VALUE

A.1.4. Communications Class (Entity Level)

The OMSC Communications class aligns with the AICDM TELECOMMUNICATIONS NETWORK ELEMENT entity. This entity is the basis of an element of a telecommunications network. TELECOMMUNICATIONS NETWORK ELEMENT allows many-to-many associations between its instances, via the TELECOMMUNICATIONS NETWORK ELEMENT ASSOCIATION entity. This association would model inter-organization communications.

The AICDM models communications equipment, but not the act of communicating (i.e., sending messages). The AICDM therefore does not align with some methods of the Communications class.

Table A-6 lists the AICDM entities that align with the Communications class at the state level.

Table A-6. AICDM entities that align with the Communications class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
TELECOMMUNICATIONS-NETWORK-ELEMENT	• getNet() • setNet()	A MILITARY-UNIT has associated TELECOMMUNICATIONS-NETWORK-ELEMENT entities. See Figure A-6.
TELECOMMUNICATIONS-NETWORK-ELEMENT-LINK-RADIO	• getNet() • setNet()	TELECOMMUNICATIONS-NETWORK-ELEMENT-LINK-RADIO is a subtype of TELECOMMUNICATIONS-NETWORK-ELEMENT. See Figure A-6.

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
TELECOMMUNICATIONS-NETWORK-ELEMENT-ASSOCIATION	• getNet() • setNet()	A TELECOMMUNICATIONS-NETWORK-ELEMENT-ASSOCIATION forms a network of TELECOMMUNICATIONS-NETWORK-ELEMENT entities. See Figure A-6.
INFORMATION-REFERENCE	• sendMessage() • receiveMessage()	An INFORMATION-REFERENCE models a message. See Figure A-16.

The degree of Entity level alignment of the Communications class is the average of the degrees of alignment of its methods (81%).

A.1.4.1. The getNet() Method (State Level)

The following ambiguities in the description of the getNet() method limit alignment analysis:

- The OMSC does not specify the types of objects that might be modeled in a communications network.

  Presumably, the objects to be modeled include components of the unit hierarchy. They may also include platforms.

- The method's description says that it the collection of objects "capable of exchanging messages." What are the bounds of capability? Can they include foes as well as friends?

  This analysis assumes that capability is defined by the setNet() method. In other words, the objects capable of exchanging messages are exactly those added by setNet(), be they friend or foe.

- Is the intent of communications to capture electronic message exchange? Or does it include other types of signals (e.g., semaphores, written communication, or for that matter verbal communication)?

  This analysis assumes that communications model messages sent over an electronic network.

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getNet() method. The result of the getNet() method should be equivalent to the set of TELECOMMUNICATIONS-NETWORK-ELEMENT entities in the network modeled by the instance of Communications.

A.1.4.2. The setNet() Method (State Level)

The ambiguities of the getNet() method (see Section A.1.4.1) apply to the setNet() method.

This is a behavioral method. Its description states that its effect is to "add the unit to the collection of objects capable of exchanging messages." This seems to imply that each simulation has a single network. If so then the AICDM, which can model multiple communications networks, is more powerful than the OMSC.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the setNet() method. The result of invoking the setNet() method is that the unit should appear in the collection of objects returned by getNet(). However, setNet() cannot be used to model communications networks. It adds a unit to a communications network, but does not provide for defining network topology. AICDM relationships among entities define network topology (the TELECOMMUNICATIONS-NETWORK-ELEMENT-ASSOCIATION entity captures this information). In the absence of topological information, the OMSC appears capable of modeling only one network. In other words, the OMSC's model of communications is weaker than the AICDM's.

A.1.4.3. The sendMessage() Method (State Level)

This is a behavioral method. It sends a message on a net.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the sendMessage() method. The result of invoking the method can be modeled by:

- The association of an INFORMATION-REFERENCE entity to an ORGANIZATION entity. The attributes of INFORMATION-REFERENCE record the time at which the message is sent (see Figure A-16).

- The association of an ACTION-OBJECTIVE-ITEM (specifying the PERSON, ORGANIZATION, or FEATURE that is to receive the message) to the ORGANIZATION that sends the message (see Figure A-15).

However:

- If an organization possesses multiple telecommunications network, it is not clear that the AICDM can model the association of a message being sent on a particular network.

- The attribute values for AICDM actions (from the ACTION-VERB CODE attribute) do not include values for sending messages.

For these reasons, the alignment is not perfect.

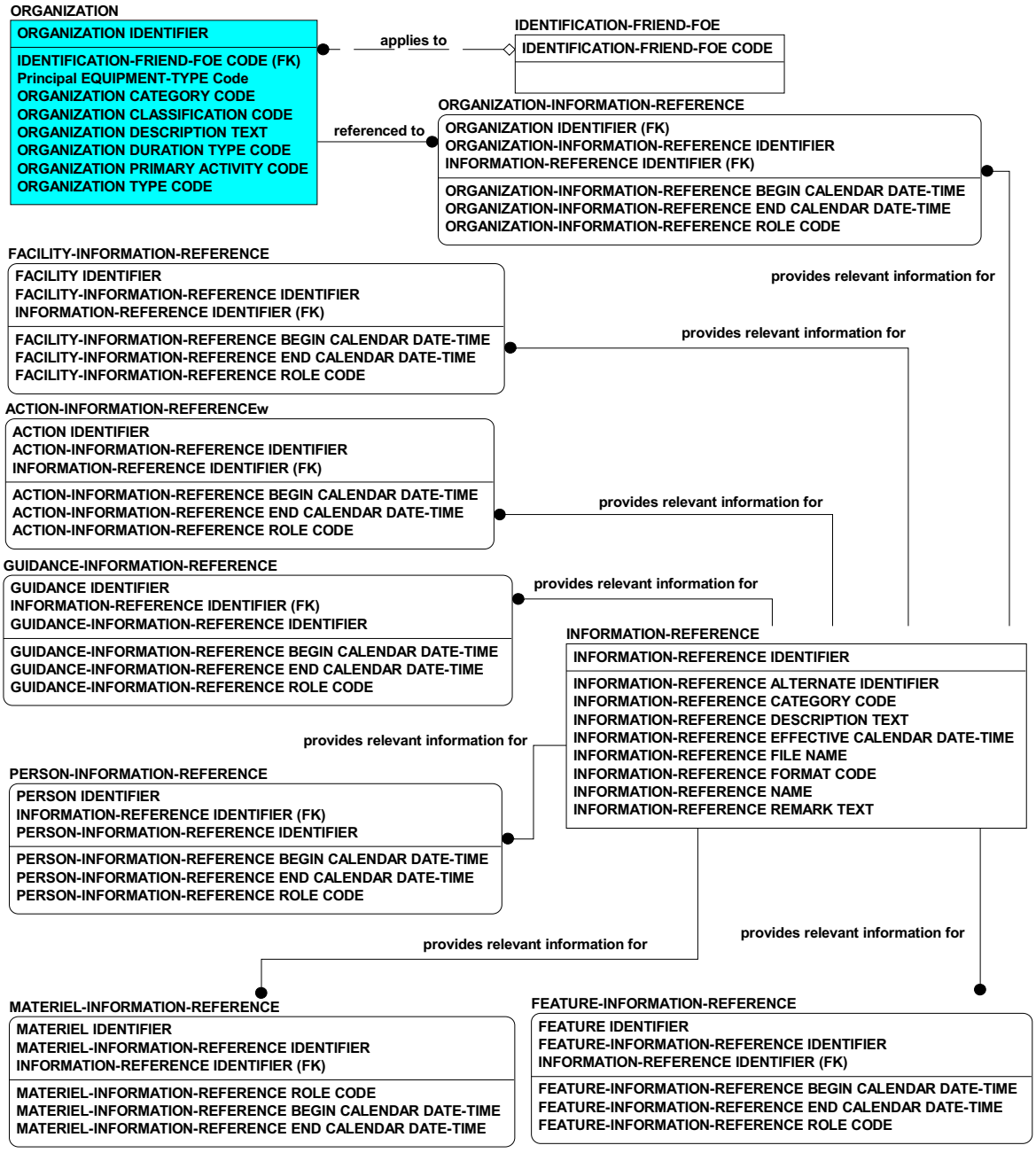


Figure A-16. AICDM Structures for Information Reference

A.1.4.4. The receiveMessage() Method (State Level)

This is a behavioral method. It receives a message on a net.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the receiveMessage() method. The result of invoking the method can be modeled by:

- The association of an INFORMATION-REFERENCE entity to an ORGANIZATION entity. The attributes of INFORMATION-REFERENCE record the time at which the message is received (see Figure A-16).

- The association of an ACTION-OBJECTIVE-ITEM (specifying the PERSON, ORGANIZATION, or FEATURE that sent the message) to the ORGANIZATION that receives the message (see Figure A-15).

However:

- If an organization possesses multiple telecommunications network, it is not clear that the AICDM can model the association of a message being received on a particular network.

- The attribute values for AICDM actions (from the ACTION-VERB CODE attribute) do not include values for sending messages.

For these reasons, the alignment is not perfect.

There is some overlap between the model elements proposed here and for the sendMessage() method in Section A.1.4.3. A data model may not need to record all the elements.

A.1.5. SystemGroup Class (Entity Level)

The SystemGroup class models systems within a unit. The descriptions of the methods associated with the class only mention types of systems, rather than individual systems. Therefore, the analysis aligns SystemGroup only to those AICDM entities that model types of systems, not individual systems. If for some reason SystemGroup must be aligned to individual systems, Table A-7 will need to include the MATERIEL entity.

Table A-7. AICDM entities that align with the SystemGroup class at the State level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
MATERIEL-ITEM and its subtypes	• getQty() • acceptLosses() • acceptGains()	A MILITARY-UNIT has associated MATERIEL-ITEM entities. See Figure A-4.
MATERIEL-ITEM-HOLDING-ESTIMATE	The MATERIEL-ITEM entity models the type of system; the MATERIEL-ITEM-HOLDING-ESTIMATE entity models the quantity.	A MILITARY-UNIT is inventoried by MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE. See Figure A-13.

The degree of Entity level alignment of the SystemGroup class is the average of the degrees of alignment of its methods (100%).

A.1.5.1. The getQty() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getQty() method. The result of getQty()maps to the value of the MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE QUANTITY attribute. See Figure A-13.

A.1.5.2. The acceptLosses() and acceptGains() Methods (State Level)

These are behavioral methods. Accepting losses (gains) means removing (adding) MATERIEL-ITEM entities.

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to these methods. The removal (addition) of a MATERIEL-ITEM entity implies the need to invoke the acceptLosses() (acceptGains()) method. The amount of losses (gains) is determined by the relation of the entity to a MATERIEL-ITEM-ORGANIZATION-HOLDING ESTIMATE entity: the losses (gains) equal the value of the MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE QUANTITY attribute.

A.1.6. Platform Class (Entity Level)

In the specification of the Unit standard object, the Platform class has no methods. Presumably it is a placeholder for the Platform standard object, whose degree of alignment is 48%. See Section A.2.

A.1.7. PlatformInfo Class (Entity Level)

In the specification of the Unit standard object, the PlatformInfo class has no methods. The class' description states that it provides information about a platform's physical characteristics.

The Platform standard object has a class named PlatformFrame. This class' getSignature() method provides physical characteristics. However, the descriptions for PlatformFrame imply that its focus is narrower than that of PlatformInfo. PlatformFrame primarily provides data used by sensors. PlatformInfo does not contain this restriction.

There is a **low degree of Entity level alignment** (25%) between the AICDM and the OMSC with respect to the PlatformInfo class. The AICDM has a limited set of entities with attributes that can model physical characteristics (see Figure A-17):

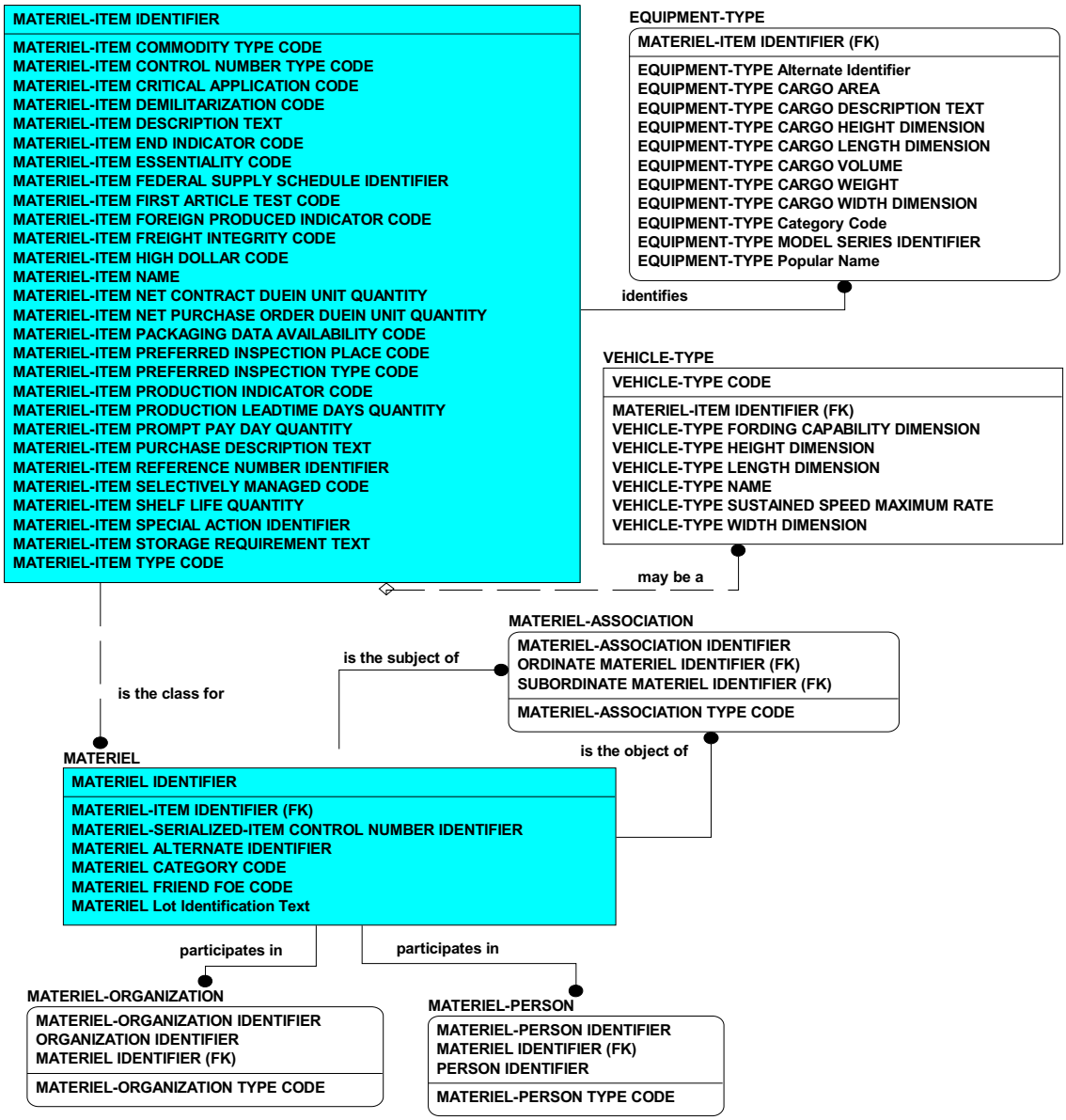


Figure A-17. AICDM Structures That May Support Platform Specifications

- Section A.2.11 discusses platform signatures.
- The AICDM can model weights and sizes of some types of equipment:
  - The EQUIPMENT-TYPE entity has height, length, width, volume, and weight attributes.
  - The VEHICLE-TYPE entity has height, length, and width attributes.[7]

[7] The DDA has other attributes that also keep track of weight, length, etc., which are not in the AICDM.

A.1.8. Attrition Class (Entity Level)

The Attrition class does not model object instances but rather encapsulates algorithms for causing attrition. The AICDM entities that align with attrition are shown in Table A-8.

Table A-8. AICDM Entities that align with the Attrition class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• MATERIEL-ITEM • PERSON-TYPE • *-ORGANIZATION-HOLDING-ESTIMATE	causeAttrition()	A MILITARY-UNIT has associated MATERIEL-ITEM entities. The association (HOLDING entities) records the number of entities. See Figure A-13.
• PERSON • PERSON-OPERATIONAL-STATUS	causeAttrition()	A MILITARY-UNIT has associated PERSON entities. A PERSON has an associated PERSON-OPERATIONAL-STATUS. See Figure A-4.
• MATERIEL • MATERIEL-OPERATIONAL-STATUS	causeAttrition()	A MILITARY-UNIT has associated MATERIEL entities. MATERIEL has an associated MATERIEL-OPERATIONAL-STATUS. See Figure A-7.

The degree of Entity level alignment of the Attrition class is the average of the degrees of alignment of its methods (75%).

A.1.8.1. The causeAttrition() Method (State Level)

The following ambiguities in the description of the causeAttrition() method limit alignment analysis:

- The OMSC does not specify how the type of attrition is determined, nor how the amount of attrition is determined. Lacking this information, the analysis is necessarily vague about the details of attrition.

Furthermore, none of the OMSC classes have any methods to specify losses. The intent is probably that losses be modeled by deleting associations. For example, the loss of a platform would be modeled by removing an existing association between a unit and a platform.

This is a behavioral method. It causes attrition to another unit. We can infer the following:

- The description of the class states that attrition is caused by means such as direct fire systems. Therefore, causing attrition requires depleting materiel.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the causeAttrition() method. The change in state caused by determineAttrition() will be losses of personnel, materiel, etc. from both the unit causing attrition and the unit receiving the damage. How an M&S system effects this depends on whether the system models individual entities or quantity of entities; e.g., whether it records that a unit has tanks identified A, B, and C or simply notes that a unit has 3 tanks. If

it models individual entities, then the change in state aligns to the following AICDM attributes (see Figure A-18):

- PERSON-ORGANIZATION BEGIN CALENDAR DATE-TIME
- PERSON-ORGANIZATION END CALENDAR DATE-TIME
- MATERIEL-ORGANIZATION (N.B. this is an entity, and it lacks a date-time attribute to record when the association exists.)
- PERSON-OPERATIONAL-STATUS CODE
- MATERIEL-OPERATIONAL-STATUS CODE

If the system models entity quantity, the change in state aligns to the following AICDM attributes:

- PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE QUANTITY,
  PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE CALENDAR DATE-TIME
- MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE QUANTITY,
  MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE CALENDAR DATE-TIME

However, the AICDM can model only limited kinds of attrition if the simulation uses entity quantity. The HOLDING entities have a STATUS CODE attribute that can record the state of a materiel item or person, but this attribute has a limited range of values. Essentially, the AICDM can model that a group of materiel items or people are in or out of action. It cannot record nuances of attrition, such as indicating that a tank is mobile but cannot use its weaponry. VALUE

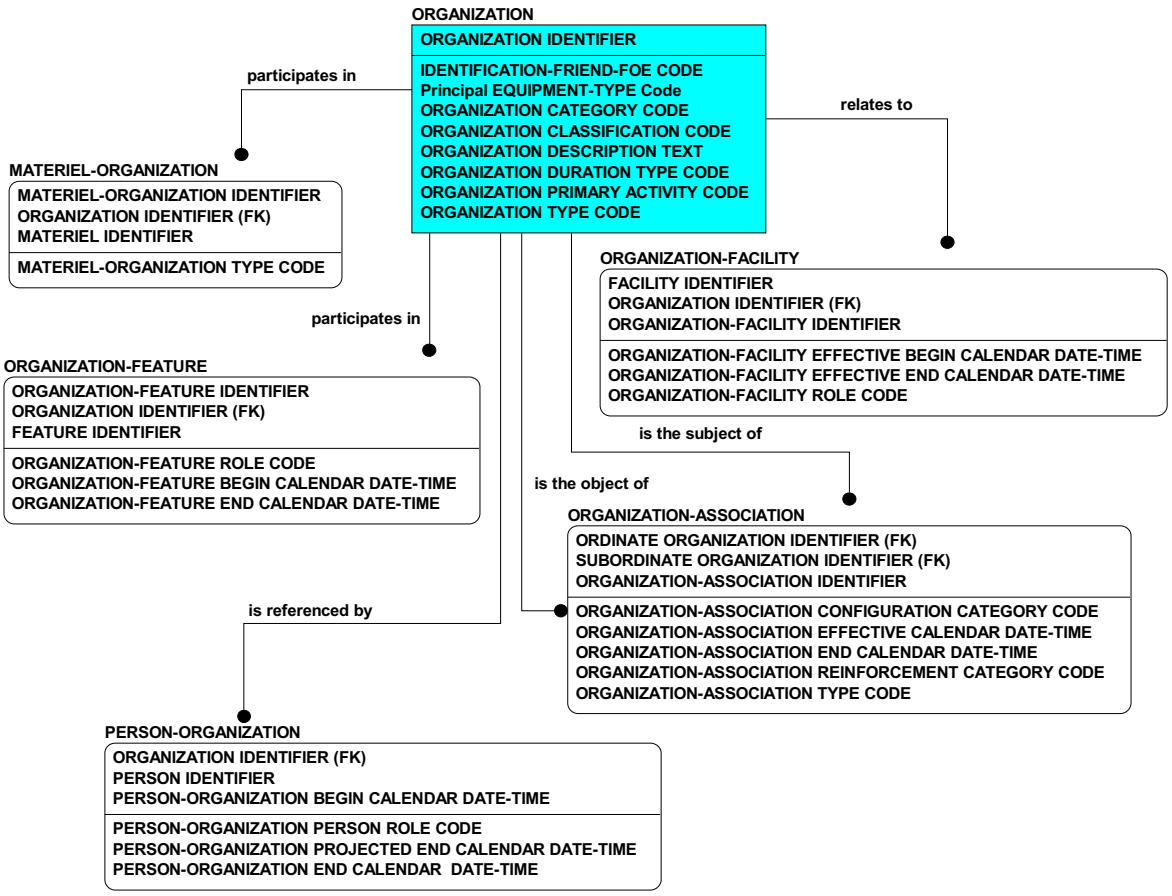


Figure A-18. AICDM Associative Entities for Organization

A.1.9. Logistics Class (Entity Level)

The Logistics class represents logistics capability of a unit. Figure A-19 shows the structures for specifying Holdings in the AICDM. Table A-9 shows the AICDM entities that align with OMSC classes.

Table A-9. AICDM entities that align with the Logistics class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE	receive()	A MILITARY-UNIT has associated MATERIEL-ITEM and PERSON-TYPE entities. The association (HOLDING entities) records the number of entities. See Figure A-19.

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• MATERIEL • PERSON • MATERIEL-OPERATIONAL-STATUS • PERSON-OPERATIONAL-STATUS	receive()	• A MILITARY-UNIT has associated MATERIEL entities. MATERIEL has an associated MATERIEL-OPTIONAL-STATUS. See Figure A-7. • A MILITARY-UNIT has associated PERSON entities. A PERSON has an associated PERSON-OPERATIONAL-STATUS. See Figure A-4.

The Logistics class description refers to types of logistics components. None of the methods in the Logistics class specify component type, so Logistics is probably an abstract class. Since Maintenance and Supply are subclasses of Logistics, the OMSC apparently intends to make use of multiple inheritance.

The degree of Entity level alignment of the Logistics class is the average of the degrees of alignment of its methods (75%).

A.1.9.1. The receive() Method (State Level)

This is a behavioral method. It increments the quantity of a logistics component.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the receive() method. As with attrition (see Section), how the effects of the receive() method align with the AICDM depend on whether the M&S system models individual logistics components or simply the quantity of types of components. If the M&S system models individual logistics components, then the effect of the method is to add associations between a Unit instance and a Logistics instance. In that case, the effect of the method in AICDM terms is to add a MATERIEL-ORGANIZATION entity, relating to the ORGANIZATION modeling the unit and the MATERIEL modeling the logistics component via "participates in" relationships.

If the M&S system models only quantity of logistics components, the effect of receive() in AICDM terms is to add a *-ORGANIZATION-HOLDING-ESTIMATE entity, relating a MATERIEL-ITEM modeling the type of logistics component to the ORGANIZATION receiving it.

The AICDM may not be able to model items that are received in less than perfect condition. See Section A.2.1.5.

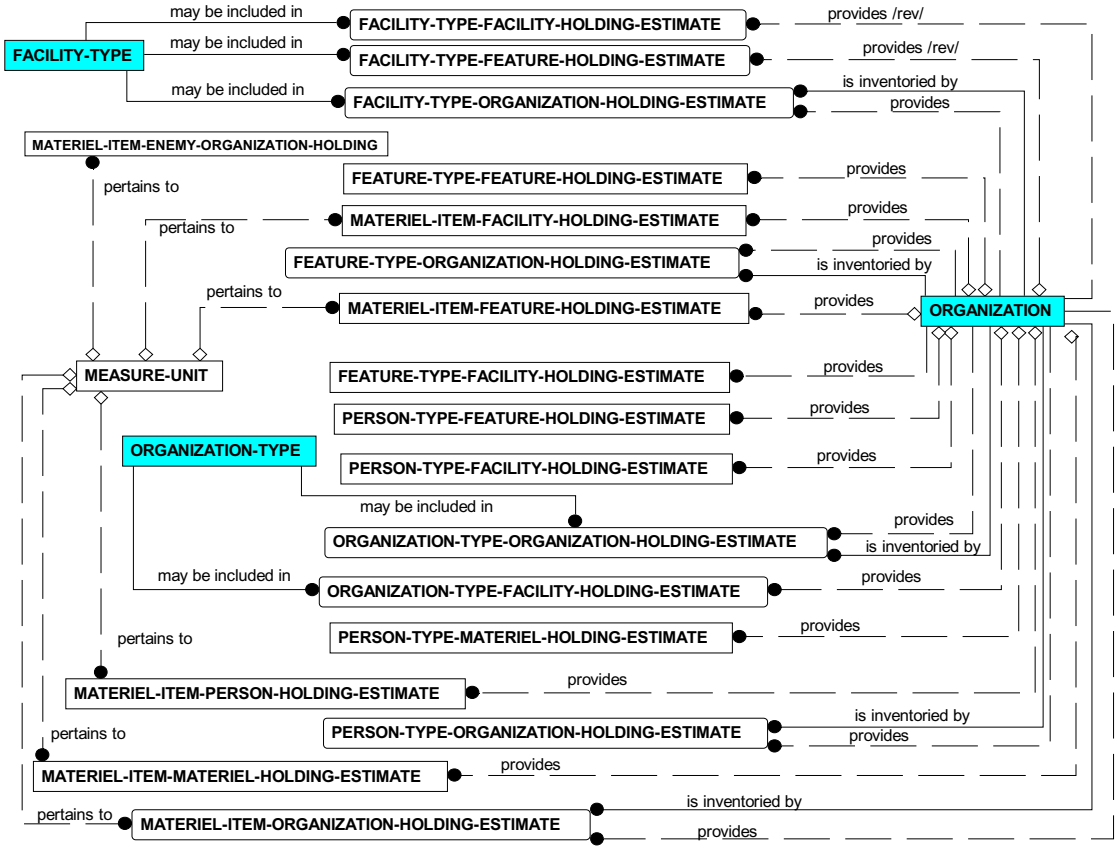


Figure A-19. AICDM Structures for Specifying Holdings

A.1.10. Maintenance Class (Entity Level)

The MAINTENANCE class represents the maintenance capability of a unit. The entities in Table A-10 identify the AICDM entities that align with the Maintenance class:

Table A-10. AICDM entities that align with the Maintenance class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• MATERIEL • MATERIEL-OPERATIONAL-STATUS • PERSON • PERSON-OPERATIONAL-STATUS	conductMaintenance() (when the M&S system is modeling individual persons and materiel items)	• A MILITARY-UNIT has associated MATERIEL entities. MATERIEL has an associated MATERIEL-OPTIONAL-STATUS. See Figure A-7. • A MILITARY-UNIT has associated PERSON entities. A PERSON has an associated PERSON-OPERATIONAL-STATUS. See Figure A-4.

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• PERSON • LOCATION • PERSON-OPERATIONAL-STATUS	• conductRecovery() • conductEvacuation() (when the M&S system is modeling individual persons)	A MILITARY-UNIT has associated PERSON entities. A PERSON has a LOCATION. See Figure A-4.
• MATERIEL • LOCATION • MATERIEL-OPERATIONAL-STATUS	• conductRecovery() • conductEvacuation() (when the M&S system is modeling individual materiel items)	A MILITARY-UNIT has associated MATERIEL. MATERIEL has a LOCATION, identified by a CONTROL-FEATURE. See Figure A-7.
• PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE	conductEvacuation() (when an M&S system is modeling number of persons or materiel items)	A MILITARY-UNIT has associated MATERIEL-ITEM and PERSON-TYPE entities. The association (HOLDING entities) records the number of entities. See Figure A-4.

The degree of Entity level alignment of the Maintenance class is the average of the degrees of alignment of its methods (42%).

A.1.10.1. The conductMaintenance() Method (State Level)

The following ambiguities in the conductMaintenance() method limit the alignment analysis:

- The OMSC does not specify how maintenance actions or medical treatment are stated. Presumably, maintenance actions and medical treatments consume resources and personnel.

- The OMSC does not specify resultant states. In other words, the effects of maintenance are not described.

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the conductMaintenance() method. Whether the AICDM can model the result of the method at all depends on whether maintenance is conducted on individual items or on groups of items. If it is conducted on individual items, then the *-OPERATIONAL-STATUS entities can be used, though they are still not fully adequate. For example (see Figure A-20):

- PERSON-OPERATIONAL-STATUS has an attribute PERSON-OPERATIONAL-STATUS CODE VALUE that captures the status of an individual. It has only a few values (e.g., "incapacitated, not walking" vs. "incapacitated, walking"). These values may or may not suffice to capture the possible states of an individual that the Maintenance class expects.

- MATERIEL-OPERATIONAL-STATUS has an attribute MATERIEL-OPERATIONAL-STATUS CODE VALUE that captures the status of equipment. It has 16 different values. These values may or may not suffice to capture the possible states of equipment that the Maintenance class expects.

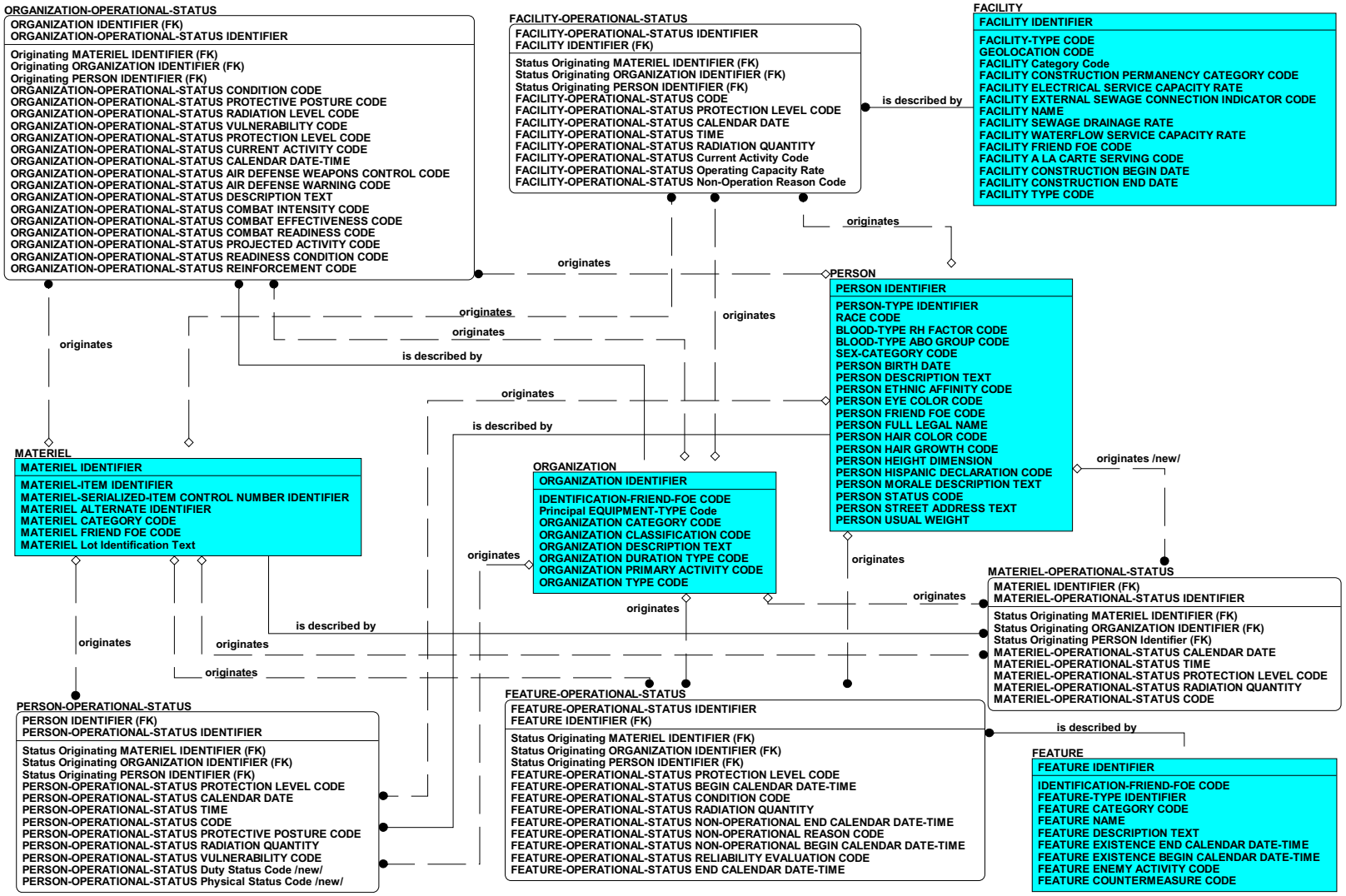


Figure A-20. AICDM Structures for Specifying Operational Status

If maintenance is conducted on groups of items, the AICDM cannot model maintenance. The holding entities apply to an entire organization. The AICDM can record a status value for all entities of a particular type, but cannot (for example) directly model a situation where half of the entities held require maintenance. Such aggregate status for a group would have to be calculated from the operational status of the individual entities' composing a group.

A.1.10.2. The conductRecovery() Method (State Level)

The following ambiguities in the conductRecovery() method limit the alignment analysis:

- The OMSC does not specify how recovery actions are stated. Presumably, recovery consumes resources and personnel.

This is a behavioral method. Its effect is to move personnel and equipment from the immediate battle area to front line medical and repair facilities.

The method's effects can be broken down more precisely to include:

- The use of personnel and consumption of materiel to recover the wounded personnel and damaged equipment.

- The change of location for the recovered personnel and the damaged equipment.

Because the changes in location are still within the front lines, we assume that invoking method does not change the unit's geometry. (It will change the unit's location, if location is computed as center of mass.)

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the conductRecovery() method. Whether the effects of the method can be modeled in the AICDM depends on whether the M&S system models individual personnel and materiel or quantities of personnel and materiel. If the system models individual personnel and materiel, then the AICDM can model changes caused by the method quite well. Invoking the method corresponds to the following changes to an AICDM model:

- Creating new PERSON-LOCATION and MATERIEL-LOCATION entities that model the new locations, and associating them with the moved personnel and materiel.

- Decrementing the supplies needed to support the recovery operation (see Section A.1.9).

- Changing the following attributes of the respective *-OPERATIONAL-STATUS entities:

    - *-CODE, which describes the condition of a person or materiel (ready vs. out of action, e.g.).

– *-PROTECTION LEVEL CODE, which describes level of vulnerability of a person or materiel. (ABOVE GROUND and BUNKERED for materiel). VALUE

If the M&S system models quantities, then the AICDM cannot model the effect of the method. The AICDM cannot record a new location for a quantity of materiel or persons.

A.1.10.3. The conductEvacuation() Method (State Level)

The following ambiguities in the conductEvacuation() method limit the alignment analysis:

- The OMSC does not specify how evacuation actions are stated. Presumably, evacuation consumes resources and personnel.

This is a behavioral method. Its effect is to move personnel and equipment to rear areas.

The method's effects can be broken down more precisely to include:

- The use of personnel and consumption of materiel to evacuate the wounded personnel and damaged equipment.

- The change of location for the evacuated personnel and the damaged equipment.

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the conductEvacuation() method. Whether the effects of the method can be modeled in the AICDM depends on whether the M&S system models individual personnel and materiel or quantities of personnel and materiel. If the system models individual personnel and materiel, then the AICDM can model the method's effects quite well (see Figure A-21). Invoking the method corresponds to the following changes to an AICDM model:

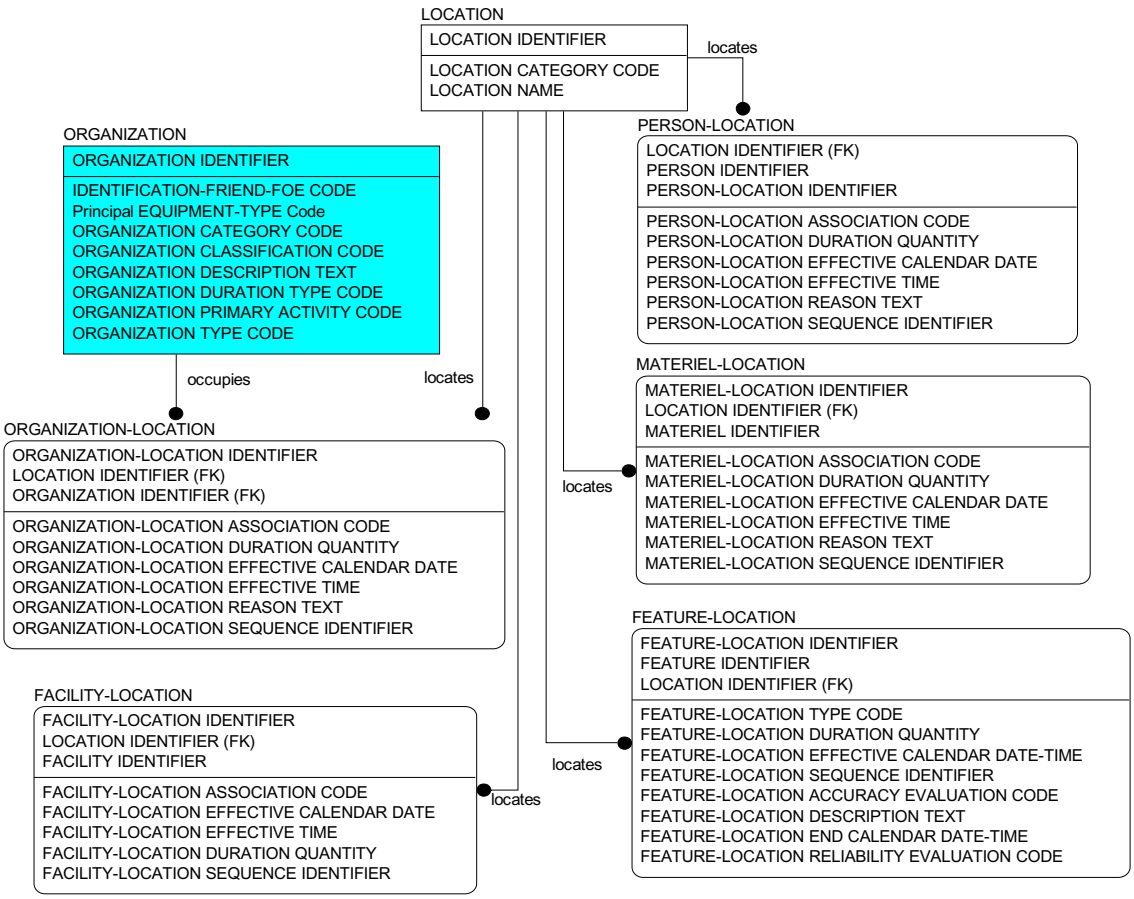


Figure A-21. AICDM Structures for Specifying Battlefield Object Location

- Creating new PERSON-LOCATION and MATERIEL-LOCATION entities that model the new locations, and associating them with the moved personnel and materiel.

- Decrementing the supplies needed to support the recovery operation (see Section A.1.9).

- Changing the following attributes of the respective *-OPERATIONAL-STATUS entities:

  – *-CODE, which describes the condition of a person or materiel (ready vs. out of action, e.g.).

  – *-PROTECTION LEVEL CODE, which describes level of vulnerability of a person or materiel. (ABOVE GROUND and BUNKERED for materiel). VALUE

If the M&S system models quantities, then the AICDM cannot model the effect of the method. The AICDM cannot record a new location for a quantity of materiel or persons.

A.1.11. Supply Class (Entity Level)

Table A-11. AICDM entities that align with the Supply class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• PERSON-TYPE-ORGANIZATION-HOLDING-ESTIMATE • MATERIEL-ITEM-ORGANIZATION-HOLDING-ESTIMATE • PERSON • MATERIEL • PERSON-OPERATIONAL-STATUS • MATERIEL-OPERATIONAL-STATUS • ORGANIZATION-TYPE-CAPABILITY-NORM • ORGANIZATION-CAPABILITY-ESTIMATE	• getRemainingCapacity() • getTotalCapacity() • getQtyOnHand() • expend() • transfer()	A MILITARY-UNIT has associated MATERIEL-ITEM and PERSON-TYPE entities. The association (HOLDING entities) records the number of entities. See Figure A-19. A MILITARY-UNIT has associated PERSON and MATERIEL entities. Each has an associated operational status. See Figure A-20. A MILITARY-UNIT has associated CAPABILITY-ESTIMATE entities that can describe maximum capacity. A MILITARY-UNIT also has an associated ORGANIZATION-TYPE, which has associated ORGANIZATION-TYPE-CAPABILITY-NORM entities that can describe capacity. See Figure A-22.

The degree of Entity level alignment of the Supply class is the average of the degrees of alignment of its methods (75%).

A.1.11.1. The getRemainingCapacity() Method (State Level)

The following ambiguities in getRemainingCapacity() limit the alignment analysis:

- The OMSC does not describe how capacity is expressed. It is therefore impossible to determine if the AICDM has an attribute that captures capacity as defined by the OMSC.

  This analysis assumes that capacity is expressed in the form used by AICDM attributes, where possible. If that does not always turn out to be true, alignment can still be implemented automatically using a translation program.

This method aligns indirectly to the AICDM. Its value would be computed as the difference between getTotalCapacity() and getQtyOnHand(). Its degree of alignment is therefore the minimum of the degrees of alignment of these two methods (75%).

A.1.11.2. The getTotalCapacity() Method (State Level)

The following ambiguities in getTotalCapacity() limit the alignment analysis:

- The OMSC does not describe how capacity is expressed. It is therefore impossible to determine if the AICDM has an attribute that captures capacity as defined by the OMSC.

  This analysis assumes that capacity is expressed in the form used by AICDM attributes, where possible. If that does not always turn out to be true, alignment can still be implemented automatically using a translation program.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getTotalCapacity() method. The AICDM can model total capacity for an organization using two entities:

1. The ORGANIZATION-CAPABILITY-ESTIMATE entity, which models capacities for a specific organization. An ORGANIZATION-CAPABILITY-ESTIMATE is associated with a CAPABILITY. The CAPABILITY defines the type of capability. The ORGANIZATION-CAPABILITY-ESTIMATE defines the actual capacity. Using these entities, the value returned by getTotalCapacity() should be equal to the value of the ORGANIZATION-CAPABILITY-ESTIMATE MEASUREMENT UNIT QUANTITY attribute (see Figure A-22). VALUE

2. The ORGANIZATION-TYPE-CAPABILITY-NORM entity, which models capacity for a type of organization. An ORGANIZATION-TYPE-CAPABILITY-NORM is also associated with a CAPABILITY. Using these entities, the value returned by getTotalCapacity() should be equal to the value of the ORGANIZATION-TYPE-CAPABILITY-NORM MEASUREMENT UNIT QUANTITY attribute (see Figure A-22). VALUE

However, the CAPABILITY TYPE CODE values do not include categories to cover many of the capacities that an M&S system must measure. Therefore, AICDM and the OMSC cannot be considered to be in perfect alignment with respect to the getTotalCapacity() method at the State level. VALUE

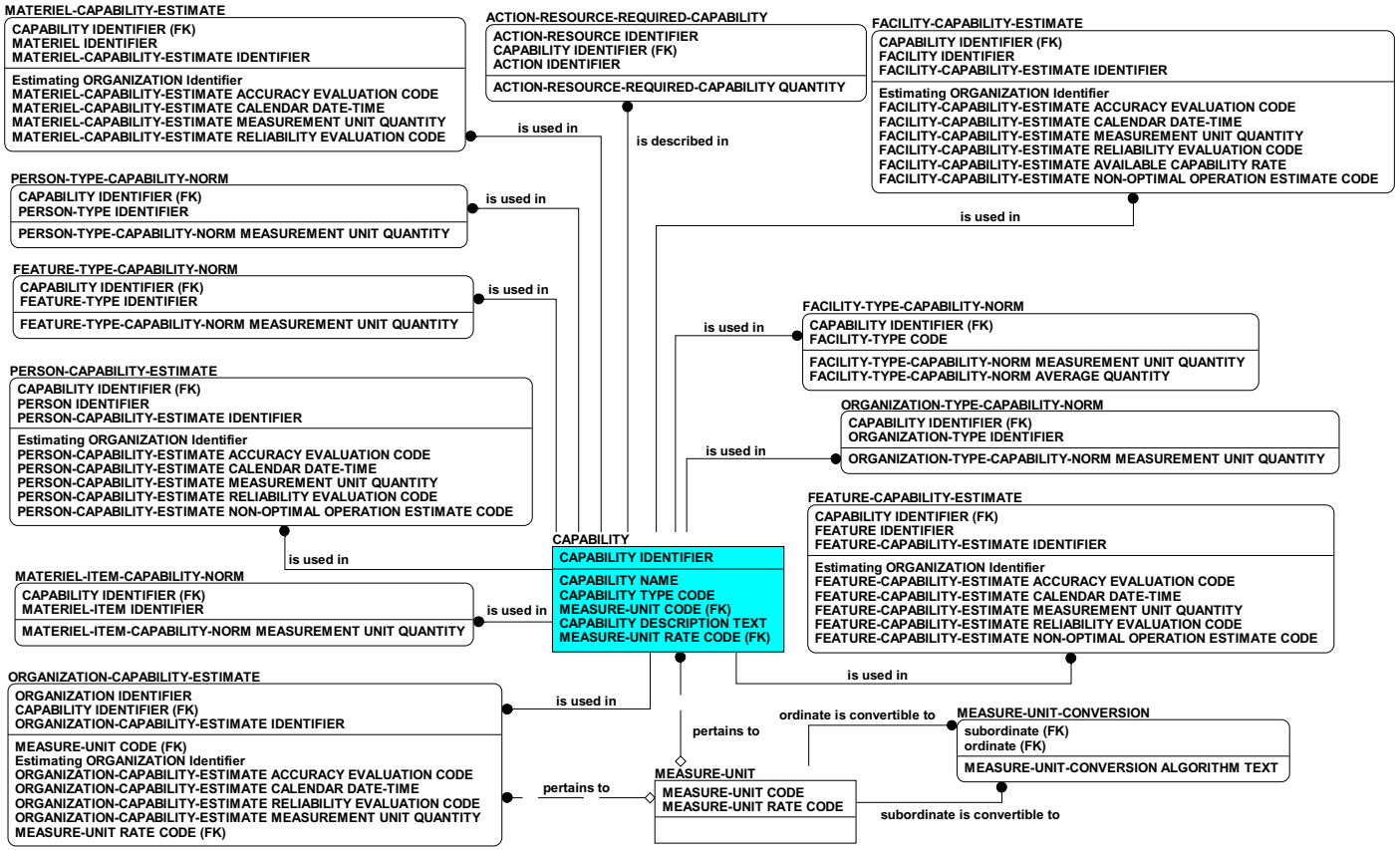


Figure A-22. AICDM Structures for Specifying Capability

A.1.11.3. The getQtyOnHand() Method (State Level)

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getQtyOnHand() method. Assuming the result of getQtyOnHand() is an integer:

- If the M&S system models entities, it should be the number of MATERIEL-ORGANIZATION (or PERSON-ORGANIZATION) instances related to the ORGANIZATION in question. (Note that PERSON-ORGANIZATION includes a timestamp attribute. This attribute would qualify the instances counted in the relationship. MATERIEL-ORGANIZATION has no such attribute. For this reason, State level alignment is not perfect.)

- If the M&S system models entity quantity, it should be the value of the appropriate HOLDING-ESTIMATE QUANTITY attribute.

In most M&S systems, the result of getQtyOnHand() will be an integer. Occasionally, quantity will be expressed in non-integer values. The AICDM cannot model these quantities. This is another reason why State level alignment is not perfect.

A.1.11.4. The expend() Method (State Level)

This is a behavioral method. It should align to the same attributes as getQtyOnHand() (75%). The difference in values returned by getQtyOnHand() before and after expend() is invoked should equal the amount specified to be transferred by the parameters to expend().In other words, this method aligns to the OMSC to the same degree that getQtyOnHand() does.

A.1.11.5. The transfer() Method (State Level)

The transfer() method is a behavioral method. It should align to the same attributes as getQtyOnHand(). The difference in values returned by getQtyOnHand() before and after transfer() is invoked should equal the amount specified to be transferred by the parameters to transfer(), both for the sender and the receiver. In other words, this method aligns to the OMSC to the same degree that getQtyOnHand() does (75%).

A.1.12. C2 Class (Entity Level)

Table A-12. AICDM entities that align with the C2 class at the entity level

AICDM Entity	Suggested By OMSC Method	Relation to MILITARY-UNIT
• ACTION • PLAN • MISSION • TASK	doC2()	A MILITARY-UNIT has associated ACTION, MISSION, and PLAN entities. A MISSION consists of TASK entities. See Figure A-12.

The degree of Entity level alignment of the C2 class is the average of the degrees of alignment of its methods (0%).

A.1.12.1. The doC2() Method (State Level)

The description of the doC2() method is highly generic. It is impossible to determine anything about the State level at this time. There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the doC2() Method.

A.2 Platform Standard Object (Conceptual Level)

The OMSC defines Platform as follows [HB 1998B]:

> A platform can be any entity of interest in the model. Examples include vehicles of all types, individuals/persons, individual systems (i.e., radar systems), a missile, etc.

The class Platform is the focal class of the Platform standard object.

The AICDM does not contain any single entity that corresponds to the OMSC concept of a platform. It does contain many entities capturing different aspects of different types of platforms.[8] The MATERIEL and AIRCRAFT entities can model vehicles. The PERSON entity can model persons. The FACILITY entity in the AICDM can model non-mobile platforms (e.g., missile silos).

Since different entities in the AICDM map to different platforms, there is no single focal entity in the AICDM for the OMSC concept of Platform. Instead, there are four focal entities: MATERIEL, AIRCRAFT, FACILITY, and PERSON.

Table A-13 contains those AICDM entities and relationships that best capture the OMSC concept of a platform. For each OMSC class, Table A-13 has four rows, one for each focal entity. The relationships between the four focal entities and relevant related entities are shown in the ER diagrams in Figure A-23 and Figure A-24.

Many of these entities and relationships come from parts of the AICDM that are marked as "to be worked" by the Army Data Management Group at Fort Belvoir. Hence, the mappings below for the Platform standard object are tentative, pending further work on the AICDM. Substantive changes are possible.

---

[8] A proposal to add entities describing platform types to the AICDM is under consideration by the Army Data Management Group at Fort Belvoir.

Table A-13. AICDM Entities that Align with Classes of the Platform Standard Object at the Conceptual Level

OMSC Class	Related AICDM Entity	Relationship to Corresponding Focal Entity
Platform (ground, water, space platforms)	MATERIEL[9]	Identical to focal entity.
Platform (air platform)	AIRCRAFT	Identical to focal entity.
Platform (person platform)	PERSON	Identical to focal entity.
Platform (static ground platform)	FACILITY	Identical to focal entity.
Sensor (ground, water, and space platforms)	MATERIEL, categorized as SENSOR-TYPE	MATERIEL has associated MATERIEL, describing a vehicle platform's sensors. MATERIEL has an associated MATERIEL-ITEM that describes its type. SENSOR-TYPE is a subtype of MATERIEL-ITEM. See Figure A-23.
Sensor (air platform)	None.	The AICDM does not associate sensors with aircraft.
Sensor (person platform)	MATERIEL, categorized as SENSOR-TYPE	PERSON has associated MATERIEL, describing a person's sensors. MATERIEL has an associated MATERIEL-ITEM, of which SENSOR-TYPE is a subtype. See Figure A-23.
Sensor (static platform)	MATERIEL, categorized as SENSOR-TYPE	FACILITY has associated MATERIEL, describing a static platform's sensors. MATERIEL has an associated MATERIEL-ITEM, of which SENSOR-TYPE is a subtype. See Figure A-24.
Weapon (ground, water, and space platforms)	MATERIEL, categorized as WEAPON-TYPE	MATERIEL has an associated MATERIEL-ITEM that describes its type. WEAPON-TYPE is a subtype of MATERIEL-ITEM. See Figure A-23.
Weapon (air platform)	None.	The AICDM does not associate weapons with individual aircraft.
Weapon (person platform)	MATERIEL, categorized as WEAPON-TYPE	PERSON has associated MATERIEL, describing a person's weapons MATERIEL has an associated MATERIEL-ITEM, of which WEAPON-TYPE is a subtype. See Figure A-23.
Weapon (static platform)	MATERIEL, categorized as WEAPON-TYPE	FACILITY has associated MATERIEL, describing a static platform's weapons. MATERIEL has an associated MATERIEL-ITEM, of which WEAPON-TYPE is a subtype. See Figure A-24.

[9] The DDA contains separate entities for SHIP and SATELLITE. We recommend including these entities in the AICDM to address M&S requirements for water and space platforms.

OMSC Class	Related AICDM Entity	Relationship to Corresponding Focal Entity
Movement (ground, water, space platforms)	LOCATION	MATERIEL can have an associated LOCATION. See Figure A-23.
Movement (person platform)	LOCATION	PERSON can have an associated LOCATION. See Figure A-23.
Movement (air platforms)	LOCATION	The AICDM does not associate a location with an aircraft.
Movement (static platforms)	N/A	A static platform is incapable of motion.
• Logistics • Supply (ground, water, and space platforms)	MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE	MATERIEL is inventoried by MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE. See Figure A-23.
• Logistics • Supply (air platform)	None	The AICDM does not associate logistical information with aircraft.
• Logistics • supply (person platform)	MATERIEL-ITEM-PERSON-HOLDING-ESTIMATE	PERSON is inventoried by MATERIEL-ITEM-PERSON-HOLDING-ESTIMATE. See Figure A-23.
• Logistics • Supply (static platforms)	MATERIEL-ITEM-FACILITY-HOLDING-ESTIMATE	FACILITY is inventoried by MATERIEL-ITEM-FACILITY-HOLDING-ESTIMATE. See Figure A-24.
Maintenance (ground, water, and space platforms)	• MATERIEL-CAPABILITY-ESTIMATE • MATERIEL-OPERATIONAL-STATUS	MATERIEL is the subject of MATERIEL-CAPABILITY-ESTIMATE. MATERIEL is described by MATERIEL-OPERATIONAL-STATUS. See Figure A-23.
Maintenance (air platforms)	None	The AICDM does not associate maintenance information with aircraft.
Maintenance (person platforms)	• PERSON-CAPABILITY-ESTIMATE • PERSON-OPERATIONAL-STATUS	PERSON is the subject of PERSON-CAPABILITY-ESTIMATE. PERSON is described by PERSON-OPERATIONAL-STATUS. See Figure A-23.
Maintenance (static platforms)	• FACILITY-CAPABILITY-ESTIMATE • FACILITY-OPERATIONAL-STATUS	FACILITY is the subject of FACILITY-CAPABILITY-ESTIMATE. FACILITY is described by FACILITY-OPERATIONAL-STATUS. See Figure A-24.
Crew (ground, water, and space platforms)	• PERSON • PERSON-TYPE • PERSON-TYPE-MATERIEL-HOLDING-ESTIMATE	A MATERIEL platform has: • Associated PERSON entities. • An associated PERSON-TYPE-MATERIEL-HOLDING-ESTIMATE. PERSON-TYPE is the type for PERSON. See Figure A-23.
Crew (air platforms)	None.	The AICDM does not associate crew information with aircraft.

OMSC Class	Related AICDM Entity	Relationship to Corresponding Focal Entity
Crew (person platforms)	N/A	N/A
Crew (static platforms)	• PERSON • PERSON-TYPE • PERSON-TYPE-FACILITY-HOLDING-ESTIMATE	A FACILITY platform has: • Associated PERSON entities. • An associated PERSON-TYPE-FACILITY-HOLDING-ESTIMATE. A PERSON has an associated PERSON-TYPE. See Figure A-24.
Communications	TELECOMMUNICATIONS-NETWORK-ELEMENT	The AICDM does not relate a TELECOMMUNICATIONS-NETWORK-ELEMENT to any of the focal entities for this view.
Carrier (ground, water, and space platforms)	• MATERIEL • MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE • MATERIEL-ITEM • VEHICLE-TYPE • PERSON • PERSON-TYPE-MATERIEL-HOLDING-ESTIMATE • PERSON-TYPE	MATERIEL has: • Associated MATERIEL and PERSON entities, for associating instances. • Associated MATERIEL-ITEM and PERSON-TYPE entities, for associating types. MATERIEL-ITEM identifies EQUIPMENT-TYPE and VEHICLE-TYPE. See Figure A-23.
Carrier (person platform)	• MATERIEL • MATERIEL-ITEM-PERSON-HOLDING-ESTIMATE • MATERIEL-ITEM • VEHICLE-TYPE • PERSON • PERSON-TYPE	PERSON has: • Associated MATERIEL and PERSON entities, for associating instances. • Associated MATERIEL-ITEM entities, for associating types. MATERIEL-ITEM identifies VEHICLE-TYPE. See Figure A-23.
Carrier (air platform)	None.	The AICDM does not model relationships between AIRCRAFT and materiel or persons.
Carrier (static platform)	N/A	Static platforms are not carriers.
PlatformFrame (ground, water, and space platforms)	• EQUIPMENT-TYPE • VEHICLE-TYPE	MATERIEL has associated MATERIEL-ITEM entities. MATERIEL-ITEM identifies EQUIPMENT-TYPE and VEHICLE-TYPE. See Figure A-23.
PlatformFrame (air platforms)	AIRCRAFT categorized as AIRCRAFT-TYPE	AIRCRAFT-TYPE categorizes AIRCRAFT.
PlatformFrame (person platforms)	PERSON	Identical to focal entity.
PlatformFrame (static platforms)	None.	The AICDM does not contain relevant information for facilities.
FrameComponent (ground, water, and space platforms)	MATERIEL, categorized as EQUIPMENT-TYPE	MATERIEL has associated MATERIEL-ITEM entities MATERIEL-ITEM identifies EQUIPMENT-TYPE. See Figure A-23.

OMSC Class	Related AICDM Entity	Relationship to Corresponding Focal Entity
FrameComponent (air platforms)	None.	The AICDM does not model aircraft component relationships.
FrameComponent (person platforms)	N/A	A person is considered atomic.
FrameComponent (static platforms)	MATERIEL, categorized as EQUIPMENT-TYPE	FACILITY has associated MATERIEL-ITEM entities. MATERIEL-ITEM identifies EQUIPMENT-TYPE. See Figure A-24.

The degree of Conceptual level alignment of the Platform standard object is the average of the degrees of alignment of its classes (48%).

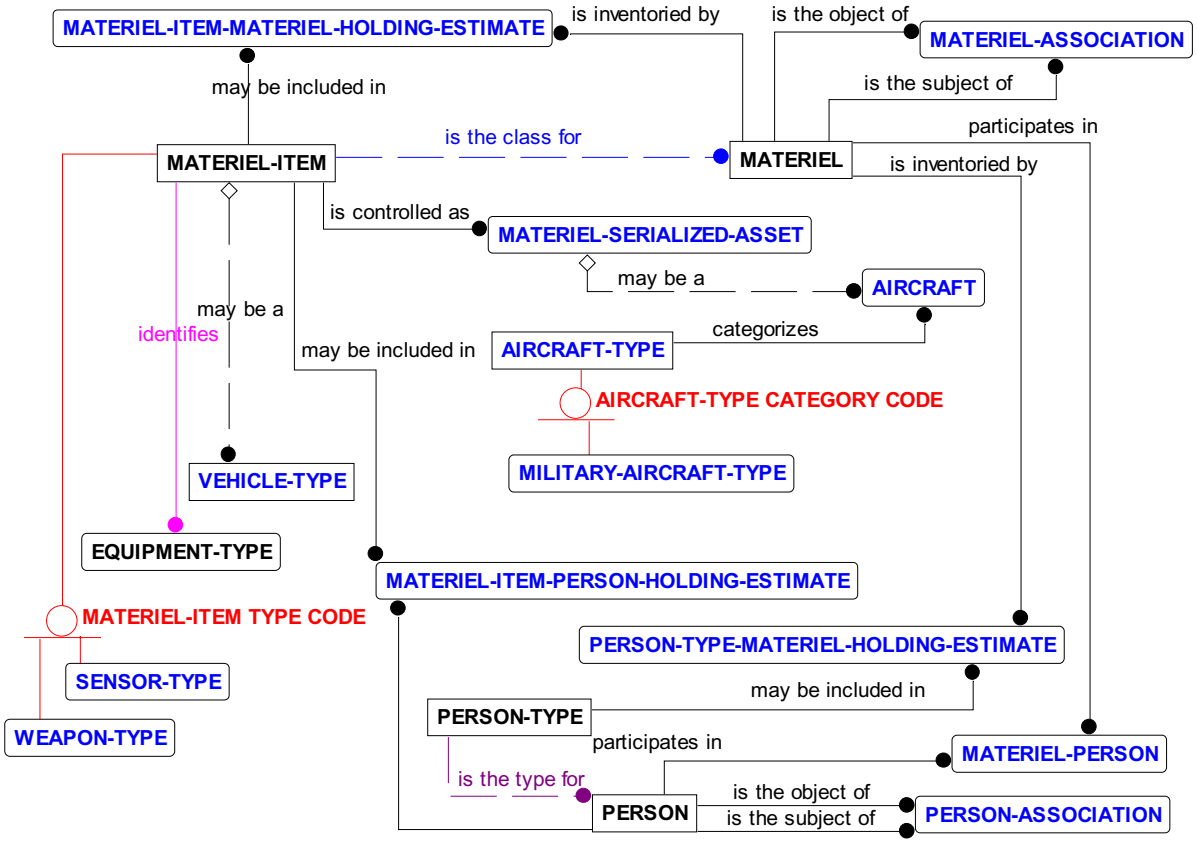


Figure A-23. Relations for Ground, Water, Space, and Person Platforms

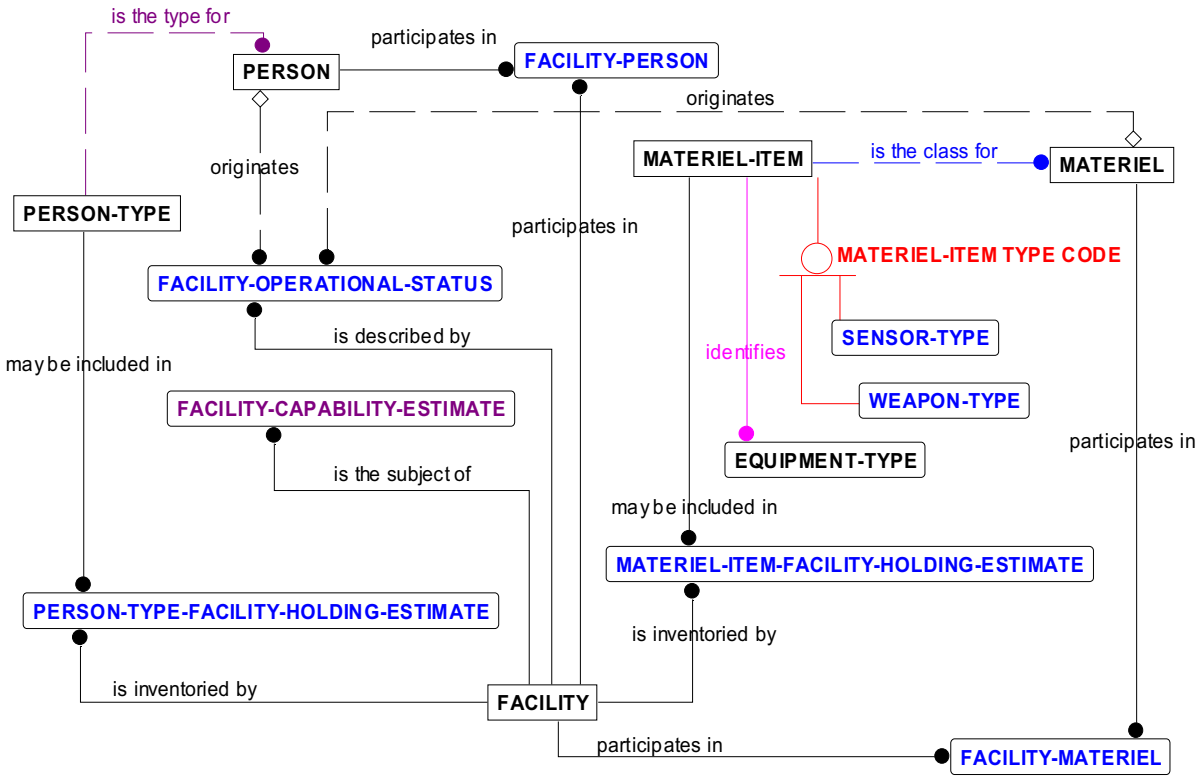


Figure A-24. Relations for Static Platforms

A.2.1. Platform Class (Entity Level)

The Platform class can model either a person or an individual system. If it is a person, it aligns with the AICDM entity PERSON. If it is an individual system, it aligns with:

- MATERIEL for ground, water, and space platforms.
- AIRCRAFT for air platforms.
- FACILITY for static platforms.

The names of the methods in class Platform suggest that the AICDM entities in Table A-14 align with Platform at the entity level, as indicated.

Table A-14. AICDM entities that align to the Platform class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to Focal Entity
MATERIEL (for ground, water, and space platforms)	N/A	Identical to focal entity.
PERSON (for person platforms)	N/A	Identical to focal entity.
AIRCRAFT (for air platforms)	N/A	Identical to focal entity.

AICDM Entity	Suggested By OMSC Method	Relation to Focal Entity
FACILITY (for static platforms)	N/A	Identical to focal entity.
MATERIEL-ITEM and its subtypes (for ground, water, and space platforms)	getType()	MATERIEL-ITEM is the class for MATERIEL
PERSON-TYPE (for person platforms)	getType()	PERSON-TYPE is the type for PERSON
• AIRCRAFT-TYPE • MILITARY-AIRCRAFT-TYPE (for air platforms)	getType()	AIRCRAFT-TYPE categorizes AIRCRAFT; MILITARY-AIRCRAFT-TYPE is a subtype of AIRCRAFT-TYPE.
FACILITY-TYPE (for static platforms)	getType()	FACILITY-TYPE is the type for FACILITY.
MATERIEL-OPERATIONAL-STATUS (for ground, water, and space platforms)	getStatus()	MATERIEL originates MATERIEL-OPERATIONAL-STATUS.
PERSON-OPERATIONAL-STATUS (for person platforms)	getStatus()	PERSON originates PERSON-OPERATIONAL-STATUS.
FACILITY-OPERATIONAL-STATUS	getStatus()	FACILITY is described by FACILITY-OPERATIONAL-STATUS.
The LOCATION view	getLocation()	A MATERIEL, PERSON, or FACILITY has an associated CONTROL-FEATURE, which has an associated LOCATION.
• ORGANIZATION • ORGANIZATION-ASSOCIATION (for ground, water, space, person, and static platforms)	getSide()	A MATERIEL, PERSON, or FACILITY has an associated ORGANIZATION, which has a side (see Section A.1.1.4).
• PERSON • PERSON-OPERATIONAL-STATUS • MATERIEL-OPERATIONAL-STATUS (for ground, water, space, person, and static platforms)	assessDamage()	PERSON, MATERIEL, and FACILITY have an associated OPERATIONAL-STATUS.

The degree of Entity level alignment of the Platform class is the average of the degrees of alignment of its methods (62%).

A.2.1.1. The getType() Method (State Level)

The following ambiguities in getType() limit alignment analysis:

- The OMSC does not define what a type designation is. Assuming that Platform is a virtual class and that actual platform instances would be members of subtypes of Platform, the type designation is probably just an identifying string that gives the name of the subtype.

Assume the type designation is a string.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getType() method:

- If the Platform is materiel, it can be modeled by the MATERIEL-ITEM entity's MATERIEL-ITEM TYPE CODE attribute, or perhaps the MATERIEL-ITEM NAME.
- If the Platform is a person, the type designation might be "Person". It might also be the person's rank or role. This information can come from the PERSON-TYPE CATEGORY CODE and the UNIFORMED-SERVICE-RANK CODE attributes of PERSON-TYPE.
- If the Platform is a facility, the type designation could come from either of the attributes FACILITY-TYPE CODE or FACILITY-TYPE CATEGORY CODE.

The uncertainties as to which attribute is appropriate precludes perfect alignment.

A.2.1.2. The getStatus() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getStatus() method:

- If the platform is a Person:
  - PERSON has an attribute PERSON-STATUS-CODE that can represent alive/dead values.
  - PERSON-OPERATIONAL-STATUS has an attribute PERSON-OPERATIONAL-STATUS CODE that captures status with enough values to model the standard kill categories.
- If the platform is MATERIEL, the MATERIEL-OPERATIONAL-STATUS CODE attribute captures the status of equipment with enough values to model the standard kill categories.
- If the platform is a FACILITY, the FACILITY-OPERATIONAL-STATUS CODE attribute captures the status of equipment with enough values to model the standard kill categories.

A.2.1.3. The getLocation() Method (State Level)

The getLocation() method for platforms can be considered analogous to the getLocation() method for units. These methods align to the same degree (37%). See Section A.3.

A.2.1.4. The getSide() Method (State Level)

The getSide() method for platforms can be considered analogous to the getSide() method for units. These methods align to the same degree (75%). See Section A.1.1.4.

A.2.1.5. The assessDamage() Method (State Level)

The following ambiguities in assessDamage() limit the alignment analysis:

- The OMSC does not specify how the amount of damage is determined.
- The OMSC does not specify units in which damage is measured.

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the assessDamage() method. The MATERIEL-OPERATIONAL-STATUS, PERSON-OPERATIONAL-STATUS, and FACILITY-OPERATIONAL-STATUS entities have a CODE attribute. This attribute describes the condition of the entity.

However, the domains of the CODE attributes do not capture the nuances of damage. VALUE

A.2.2. Sensor Class (Entity Level)

Table A-15. AICDM entities that align to the Sensor class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to Focal Entity
SENSOR-TYPE	getMaxRange()	The focal entities (except AIRCRAFT) have an association with MATERIEL, which has an associated MATERIEL-ITEM, of which SENSOR-TYPE is a subtype.
None	getOrientation()	N/A
PERSON	getContacts()	N/A
FACILITY	getContacts()	
MATERIEL-ITEM	getContacts()	MATERIEL-ITEM is the class for MATERIEL
ORGANIZATION	getContacts()	MATERIEL has a many-to-many relationship with ORGANIZATION: • MATERIEL is inventoried by MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE • ORGANIZATION provides MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE
None	activate()	N/A
	deactivate()	N/A

The degree of Entity level alignment of the Sensor class is the average of the degrees of alignment of its methods (30%).

A.2.2.1. The getMaxRange() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getMaxRange() method. The SENSOR-TYPE entity has an attribute SENSOR-TYPE-RANGE MAXIMUM DIMENSION that models the maximum range of a sensor type.

A.2.2.2. The getOrientation() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getOrientation() method. The AICDM does not model orientation.[10]

A.2.2.3. The getContacts() Method (State Level)

The following ambiguities in getContacts() limit the alignment analysis:

- The OMSC states that getContacts() queries a target, but does not describe the result of that query.

  Assuming that a target may be a PERSON, FACILITY, MATERIEL entity, or ORGANIZATION, the result of the getContacts() method might be a set of said entities. Perhaps other information, such as location or physical properties, might be included.

Depending on the sensor, the method may be able to distinguish friends from foes (see Section A.1.1.4 for relevant attributes).

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the getContacts() method. The estimate is based on the ambiguity described above. The AICDM does not model many physical object signatures. If the method returns signatures, it does not align to the AICDM. If it returns contacted entities (as suggested by the name) it aligns well.

A.2.2.4. The activate() and deactivate() Methods (State Level)

There is **no State level alignmen**t (0%) between the AICDM and the OMSC with respect to these methods. The AICDM entity MATERIEL-OPERATIONAL-STATUS describes the operational ability of specific materiel.

The attribute MATERIEL-OPERATIONAL-STATUS CODE has a range of values listing various VALUE abilities. These values describe capability to operate, rather than actual operation.

A.2.3. Weapon Class (Entity Level)

Table A-16. AICDM entities that align to the Weapon class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to Focal Entity
WEAPON-TYPE	getMaxRange()	A weapon is MATERIEL, which has an associated

[10] See footnote 5.

AICDM Entity	Suggested By OMSC Method	Relation to Focal Entity
		MATERIEL-ITEM; WEAPON-TYPE is a subtype of MATERIEL-ITEM.
• ACTION • ACTION-OBJECTIVE-MATERIEL • ACTION-VERB	load()	ACTION describes loading. ACTION-OBJECTIVE-MATERIEL identifies a particular weapon as the object of loading. The focal entities (MATERIEL, PERSON, etc.) have associations with weapon MATERIEL.
• ACTION • ACTION-RESOURCE-MATERIEL • TARGET • ACTION-VERB	engageTarget()	An ACTION: • Is employed against a TARGET. • Uses ACTION-RESOURCE-MATERIEL (i.e., the weapon). A TARGET has an association with the entity (ORGANIZATION, MATERIEL, or PERSON) that designated it as a target.

The degree of Entity level alignment of the Weapon class is the average of the degrees of alignment of its methods (50%).

A.2.3.1. The getMaxRange() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getMaxRange() method. The result of getMaxRange() should equal the value of the WEAPON-TYPE-RANGE MAXIMUM DIMENSION attribute. Possibly a simulation would use the WEAPON-TYPE RANGE MAXIMUM ASSISTED DIMENSION attribute, if the weapon is on a carrier.

A.2.3.2. The load() Method (State Level)

There is **a low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the load() method. The AICDM can model the action of loading a weapon, but it does not model whether a weapon is loaded or not. Arguably, a weapon's state could be ascertained based on an examination of recorded loading and firing actions, but such an approach seems complex.

ACTION-VERB CODE lacks a value for weapon loading. One should be added to support M&S system modeling. VALUE

A.2.3.3. The engageTarget() Method (State Level)

The following ambiguities in engageTarget() limit the alignment analysis:

- The method's description states that it initiates the weapon-firing event. This might imply that weapon firing is not an atomic event. The OMSC does not break down weapon firing further.

This is a behavioral method. The AICDM entity ACTION can model the action of firing a weapon. Its attribute ACTION-VERB CODE includes domain values for weapon firing.

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the engageTarget() method. Each engagement of a target corresponds to an ACTION. The AICDM models the target as a TARGET (a indirect subtype of an ACTION-OBJECTIVE, which is associated with an ACTION). The weapon used is associated with the ACTION as ACTION-RESOURCE-MATERIEL. The TARGET is associated with the platform engaging the weapon via an "initiates" relationship to one of MATERIEL (ground, water, and space platforms), PERSON (person platforms), or FACILITY (static platforms).

However, the AICDM cannot model the result of the method using attributes of a WEAPON. In particular, the method should decrement the number of loaded munitions by one. Modeling this using the AICDM would be complex, for the same reasons stated for the load() method (see Section A.2.3.2).

ACTION-VERB CODE has several domain values for weapon firing (FIRE FOR EFFECT, VALUE PROVIDE SPREADING FIRE, etc.). None corresponds exactly to initiating weapon firing. In any case, the exact value assigned to the attribute presumably depends on context, i.e., the type of mission that Platform is supporting.

A.2.4. Movement Class (Entity Level)

The Movement class is concerned with the motion of a platform. The class' methods set and return motion-related properties.

Table A-17. AICDM entities that align to the Movement class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL or PERSON
LOCATION	moveTo()	MATERIEL has a many-to-many relationship with LOCATION. PERSON has a many-to-many relationship with LOCATION.
None	getVelocity()	N/A
None	changeVelocity()	N/A

The degree of Entity level alignment of the Movement class is the average of the degrees of alignment of its methods (25%).

A.2.4.1. The getVelocity() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getVelocity() method. The AICDM does not model velocity (see Section A.1.1.2).

A.2.4.2. The changeVelocity() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the changeVelocity() method. The AICDM does not model velocity (see Section A.1.1.2).

A.2.4.3. The moveTo() Method (State Level)

This is a behavioral method. Its behavior is similar to that of the move() method of the Unit class (see Section A.1.1.9). However, that method "advances a unit toward its next location," whereas moveTo() "move[s] directly to a location". The description of moveTo() seems more specific than that of move(), implying the parameters to moveTo() must include a Location.

The other difference is that the change of state for move() creates a new association between an ORGANIZATION and a LOCATION (via a CONTROL-FEATURE). Invoking moveTo() creates a new association between MATERIEL (or PERSON) and a LOCATION.

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the moveTo() method. As the first paragraph of this section states, moveTo() aligns more closely than Unit.move(). This method aligns more closely to the AICDM than the Unit.move() method. However, the AICDM cannot model all types of locations that the OMSC uses (see Section A.3). The alignment is therefore not perfect.

A.2.5. Logistics Class (Entity Level)

The Logistics class is identical to the Logistics class for a Unit (see Section A.1.9), except that changes affect relationships between the MATERIEL entity modeling the platform and its logistics supplies, rather than between an ORGANIZATION entity modeling a unit. The Logistics class of the Platform standard object aligns to the AICDM to the same degree as the Logistics class of the Unit standard object (75%).

A.2.6. Supply Class (Entity Level)

The Supply class is identical to the Supply class for a Unit (see Section A.1.11), except that changes affect relationships between the MATERIEL entity modeling the platform and its logistics supplies, rather than between an ORGANIZATION entity modeling a unit and its logistics supplies. The Supply class of the Platform standard object aligns to the AICDM to the same degree as the Supply class of the Unit standard object (75%).

A.2.7. Maintenance Class (Entity Level)

The Maintenance class models maintenance actions. It is similar to the Maintenance class for a Unit (see Section A.1.10), except it has only one method (conductMaintenance()) rather than three. A platform may be a person as well as equipment, so the Maintenance class can be used to describe medical treatment.

The degree of Entity level alignment of the maintenance class is the average of the degrees of alignment of its methods (25%).

A.2.7.1. The conductMaintenance() Method (State Level)

The conductMaintenance() method aligns to the AICDM to the same degree as does the conductMaintenance() method of the Unit class (25%). See Section A.1.10.1.

A.2.8. Crew Class (Entity Level)

The Crew class models the persons assigned to a platform as its crew. Presumably this platform is materiel, not a person or facility, and the crew are persons, not materiel or facilities.

Table A-18. AICDM entities that align with the Crew class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL
PERSON	getQuantity()	MATERIEL has associated PERSON entities.
PERSON-TYPE	getQuantity()	MATERIEL has associated PERSON-TYPE entities.

The degree of Entity level alignment of the Crew class is the average of the degrees of alignment of its methods (100%).

A.2.8.1. The getQuantity() Method (State Level)

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the getQuantity() method. The reasons are similar to those for the getQtyOnHand() method in Section A.1.11.3. That method had a high degree of State level alignment. The situations where it was imperfect had to do with modeling materiel quantity. The getQuantity() method always models people.

Some AICDM attributes have values that suggest crew. MATERIEL-PERSON has an attribute MATERIEL-PERSON TYPE CODE. One value of this code is "is assigned to". Another is "is operated by". If it is necessary to only record the relationship between a specific platform and the number of its crew, then the HOLDING estimate could be used. MATERIEL has a many-to-many association with PERSON-TYPE: VALUE

- MATERIEL is inventoried by PERSON-TYPE-MATERIEL-HOLDING-ESTIMATE
- PERSON-TYPE maybe included in MATERIEL-ITEM-MATERIEL-HOLDING-ESTIMATE

A.2.9. Communications Class (Entity Level)

The Communications class is identical to the Communications class for a Unit (see Section A.1.4), except that changes affect relationships between the MATERIEL entity model-

ing the platform and its communications network, rather than between an ORGANIZATION entity modeling a unit and its communications network. The Communications class of the Platform standard object aligns to the AICDM to the same degree as the Communications class of the Unit standard object (81%).

A.2.10. Carrier Class (Entity Level)

The Carrier class models the ability of a platform to carry personnel and materiel. This type of platform is MATERIEL, not a PERSON or FACILITY. A carrier cannot carry a FACILITY.

Table A-19 shows the AICDM entities that model carrying these things. It includes:

- AICDM entities that can represent the individual personnel and materiel carried by a platform.
- The holdings estimates for numbers of personnel and materiel that may be carried by a platform.
- The occurrence of load and unload activities, which can be recorded via the AICDM ACTION entity.

The value for the activity of loading or unloading materiel or persons onto a platform can be modeled via the ACTION-related entities, where ACTION-OBJECTIVE-MATERIEL could specify the platform which is being loaded or unloaded. If there is a need to specify which battlefield object must carry out this activity then the ACTION-RESOURCE-ITEM or the ACTION-RESOURCE-TYPE hierarchies can be used to capture that aspect of loading and unloading, e.g., ACTION-RESOURCE-PERSON or ACTION-RESOURCE-ORGANIZATION. Additional domain values may be required for the ACTION-VERB CODE to support this requirement.

Table A-19. AICDM entities that align with the Carrier class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL
MATERIEL (ground, water, and space)	• load() • unload()	MATERIEL has a many-to-many relationship to MATERIEL via a MATERIEL-ASSOCIATION.
PERSON (person)	• load() • unload()	MATERIEL has associated PERSON entities.
MATERIEL-ITEM (ground, water, and space)	• load() • unload() • getRemainingCapacity() • getTotalCapacity() • getQtyOnHand()	MATERIEL has associated MATERIEL-ITEM entities; the association estimates the number of items of the type.
PERSON-TYPE (person)	• load() • unload() • getRemainingCapacity() • getTotalCapacity() • getQtyOnHand()	MATERIEL has associated PERSON-TYPE entities; the association estimates the number of persons of the type.

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL
• ACTION • ACTION-OBJECTIVE and its subtypes • ACTION-RESOURCE and its subtypes	• load() • unload()	MATERIEL is specified as ACTION-OBJECTIVE-MATERIEL. MATERIEL is specified as ACTION-RESOURCE-MATERIEL.

The degree of Entity level alignment of the Carrier class is the average of the degrees of alignment of its methods (85%).

A.2.10.1. The load() Method (State Level)

The following ambiguities in load() (and unload(), below) limit the alignment analysis:

- The results of this method are not clear. Consider loading personnel onto a platform. Neither Platform nor Carrier has any associations with a class that could model personnel. Perhaps the intent of this method is to model loading a quantity of personnel or materiel onto a carrier, rather than modeling instances.

This is a behavioral method. After it is invoked, the getRemainingCapacity() method should reflect that the carrier's capacity has been diminished by the number of items loaded.

There is **perfect State level alignment** (100%) between the AICDM and the OMSC with respect to the load() method. Modeling load() in the AICDM probably (given the ambiguity) means creating a new instance of MATEREL-ITEM-MATERIEL-HOLDING-ESTIMATE or PERSON-TYPE-MATERIEL-HOLDING-ESTIMATE and associating it with the platform through the relationships listed in Table A-19.

The MATEREL-ITEM-MATERIEL-HOLDING-ESTIMATE entity has several attributes whose values can probably be fixed for the purposes of alignment VALUE

- MATEREL-ITEM-MATERIEL-HOLDING-ESTIMATE ACCURACY EVALUATION CODE denotes the accuracy of the estimate. For a simulation, the estimate can probably be considered accurate, so the value of the attribute is CONFIRMED. VALUE

- MATEREL-ITEM-MATERIEL-HOLDING-ESTIMATE RELIABILITY EVALUATION CODE denotes the degree of reliability of the estimate. For a simulation, the estimate can probably be considered reliable, so the value of the attribute is COMPLETELY RELIABLE. VALUE

- MATEREL-ITEM-MATERIEL-HOLDING-ESTIMATE TYPE CODE characterizes the estimate. For a simulation, its value can be ACTUAL. VALUE

A.2.10.2. The unload() Method (State Level)

This is a behavioral method. It aligns to the same degree that the load() method aligns.

A.2.10.3. The getRemainingCapacity() Method (State Level)

This is a behavioral method. Its effect is analogous to, and its degree of alignment equivalent to, that of the getRemainingCapacity() method of the Supply class (75%).

A.2.10.4. The getTotalCapacity() Method (State Level)

This is a behavioral method. Its effect is analogous to, and its degree of alignment equivalent to, that of the getTotalCapacity() method of the Supply class (75%).

A.2.10.5. The getQtyOnHand() Method (State Level)

This method aligns indirectly to the AICDM. Its degree of alignment is the minimum of the degrees of alignment of getRemainingCapacity() and getTotalCapacity(), from which its value is calculated (75%).

A.2.11. PlatformFrame Class (Entity Level)

The PlatformFrame class contains physical models of a platform. The OMSC describes a physical model as follows:

> This may be a detailed model, but typically is data required by sensors to acquire/detect the platform. Examples of the physical data are the visual signature, thermal signature, acoustic signature and cross sectional area. Platform orientation and other descriptions also belong here.

The AICDM contains descriptive data for a variety of physical characteristics of platforms:

- The EQUIPMENT-TYPE entity has height, length, width, volume, and weight attributes.
- The VEHICLE-TYPE entity has height, length, and width attributes.
- The PERSON entity has height, usual weight, hair color, and eye color attributes.

The PlatformFrame class contains physical models of a platform. These models typically supply data required by sensors to detect the platform. The AICDM does not generally contain such data. The exception is cross-sectional area, modeled as shown in Table A-20.

Table A-20. AICDM entities that align to the PlatformFrame class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL
EQUIPMENT-TYPE (ground, water, and space platforms)	getSignature()	MATERIEL has MATERIEL entities, for associating instances, and associated MATERIEL-ITEM entities, for associating types. MATERIEL-ITEM identifies EQUIPMENT-TYPE.
VEHICLE-TYPE (ground platforms)	getSignature()	MATERIEL has MATERIEL entities, for associating instances, and associated MATERIEL-ITEM entities, for associating types. MATERIEL-ITEM identifies VEHICLE-TYPE.
PERSON (person platforms)	getSignature()	PERSON has MATERIEL entities, for associating instances, and associated MATERIEL-ITEM entities, for associating types. MATERIEL-ITEM identifies EQUIPMENT-TYPE.

The degree of Entity level alignment of the PlatformFrame class is the average of the degrees of alignment of its methods (25%).

A.2.11.1. The getSignature() Method (State Level)

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the getSignature() method. Of the example signatures given, the AICDM can accommodate only a few, including weight and dimensions.

A.2.12. FrameComponent Class (Entity Level)

A FrameComponent provides for hierarchical structuring of a platform. Table A-21 shows the AICDM entities that provide this type of structuring. The sole method in the class provides for a signature, just as in the previous section.

Table A-21. AICDM entities that align to the FrameComponent class at the Entity level

AICDM Entity	Suggested By OMSC Method	Relation to MATERIEL
MATERIEL	getSignature()	N/A
MATERIEL-ASSOCIATION	getSignature()	MATERIEL uses MATERIEL-ASSOCIATION to achieve structure.

One value for the MATERIEL-ORGANIZATION TYPE CODE attribute is IS A COMPONENT OF. VALUE This seems to capture the intent of the FrameComponent relationships.

The degree of Entity level alignment of the FrameComponent class is the average of the degrees of alignment of its methods (25%).

A.2.12.1. The getSignature() Method (State Level)

The getSignature() method aligns to the AICDM to the same degree as the getSignature() method of the PlatformFrame class (25%). See Section A.2.11.1.

A.3 Location Standard Object (Conceptual Level)

The Location standard object is defined in a technical report dated October 1998 [JHB 1998]. The report is designated as not completed.

An OMSC location is "typically" a point. The OMSC's Unit standard object has a method UnitGeometry.getShape(). We infer from this that a location is not a shape, and therefore define a conceptual level view for the AICDM that models only points, not geometric figures.

The Location class is the focal class of the Location standard object.

POINT is the focal entity of the AICDM view for a location.

Table A-22. AICDM entities that align with classes of the Location standard object at the Conceptual level

OMSC Class	Related AICDM Entity	Relation to POINT
Geocentric	POINT	Identical (the same entity)
Geocentric	MEASURED-ELEVATION-POINT	MEASURED-ELEVATION-POINT is a subtype of POINT. See Figure A-2.

The degree of Conceptual level alignment of the Location standard object is the average of the degrees of alignment of its classes (37%).

A.3.1. Location Class (Entity Level)

The Location class does not directly align with any AICDM classes. The reason is that it is a superclass, and AICDM entities related to location do not use a corresponding superclass model. However, subclasses of Location align to AICDM entities.

The method names in Location do not indicate a need for any other AICDM entities.

The degree of Entity level alignment of the Location class is the average of the degrees of alignment of its methods (37%).

A.3.1.1. The distanceFrom() Method (State Level)

The distanceFrom() method calculates the distance from one point to another. It is either a class method requiring two parameters—both Location instances—or an instance method requiring one parameter, and using the invoking instance as the second point. It returns a

non-negative value representing the distance between the two locations. The units of the distance are unspecified.

There is a **low degree of State level alignment** (25%) between the AICDM and the OMSC with respect to the distanceFrom() method. The AICDM can model only geodetic locations and does not model distances.

A.3.1.2. The convert() Method (State Level)

The convert() method converts points from one coordinate system to another. It is either a class method requiring one parameter—a Location instance—or an instance method using the invoking instance as the point. It returns a value in the opposite coordinate system. (If the OMSC ever adds more subclasses of Location, convert() will need a different parameter schema.

There is a **medium degree of State level alignment** (50%) between the AICDM and the OMSC with respect to the convert() method. The AICDM can model only geodetic distances. The AICDM provides for conversion between coordinate systems (actually, between any two units of measure) through the MEASURE-UNIT-CONVERSION entity. Its MEASURE-UNIT-COVERSION ALGORITHM TEXT attribute provides text of an algorithm to convert from one MEASURE-UNIT to another. However, the attribute is free text, not a formal mapping (see Figure A-22).

A.3.2. Local Class (Entity Level)

The Local class models coordinates expressed in a local (Cartesian) coordinate system.

The AICDM does not model locations as Cartesian coordinates. The AICDM and the OMSC do not align with respect to the Local class at the Entity level.[11]

The degree of Entity level alignment of the Local class is the average of the degrees of alignment of its methods (0%).

A.3.2.1. The getXCoordinate() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getXCoordinate() method .

---

[11] The Army has developed structures for international operations that provide for this type of specification. The ADMG at Ft. Belvoir is considering including them in future versions of the AICDM.

A.3.2.2. The getYCoordiante() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getYCoordinate() method .

A.3.2.3. The getZCoordinate() Method (State Level)

There is **no State level alignment** (0%) between the AICDM and the OMSC with respect to the getZCoordinate() method.

A.3.3. LatLon Class (Entity Level)

The Geocentric class models coordinates expressed in a geodetic coordinate system. Table A-22 shows the AICDM entities that map to OMSC classes.

The degree of Entity level alignment of the LatLon class is the average of the degrees of alignment of its methods (75%).

A.3.3.1. The getLatitude() Method (State Level)

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getLatitude() method. The value returned by the getLatitude() method aligns with three AICDM attributes of POINT (see Figure A-14):

- The POINT LATITUDE COORDINATE attribute, which represents the actual latitude.
- The HORIZONTAL REFERENCE DATUM CODE attribute, which provides a reference system for interpreting the latitude. (Presumably the value of this attribute would be constant for all latitudes in a given M&S system.)
- The POINT HORIZONTAL PRECISION QUANTITY attribute, which specifies the precision of the latitude.

The value returned by getLatitude() is in seconds. There is no explicit discussion in the AICDM of the precision of POINT LATITUDE COORDINATE. However, the AICDM is based on WGS 84, which states that positions shall be reported precise to tenths of a second [SNF]. Therefore, getLatitude() and POINT LATITUDE COORDINATE do not align perfectly.

A.3.3.2. The getLongitude() Method (State Level)

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getLongitude() method. The value returned by the getLongitude() method aligns with three AICDM attributes of POINT (see Figure A-14):

- The POINT LONGITUDE COORDINATE attribute, which represents the actual latitude.

- The HORIZONTAL REFERENCE DATUM CODE attribute, which provides a reference system for interpreting the latitude. (Presumably the value of this attribute would be constant for all latitudes in a given M&S system.)

- The POINT HORIZONTAL PRECISION QUANTITY attribute, which specifies the precision of the latitude.

Two assumptions about OMSC latitudes and longitudes are necessary for alignment between the two models:

- Latitudes and longitudes are specified using the same horizontal reference.

- Latitudes and longitudes are specified using the same precision.

Both these assumptions seem highly probable.

The value returned by getLongitude() is in seconds. There is no explicit discussion in the AICDM of the precision of POINT LONGITUDE COORDINATE. However, the AICDM is based on WGS 84, which states that positions shall be reported precise to tenths of a second [SNF]. Therefore, getLongitude() and POINT LONGITUDE COORDINATE do not align perfectly.

A.3.3.3. The getAltitude() Method (State Level)

There is a **high degree of State level alignment** (75%) between the AICDM and the OMSC with respect to the getAltitude() method. The value returned by the getAltitude() method aligns with the following attributes of MEASURED-ELEVATION-POINT (see Figure A-14):

- MEASURED-ELEVATION-POINT ELEVATION DIMENSION, which specifies the elevation.

- MEASURED-ELEVATION-POINT PRECISION QUANTITY, which specifies the precision of the elevation.

- MEASURED-ELEVATION-POINT TYPE CODE, which provides a reference system for interpreting the elevation.

The value returned by getAltitude() is in feet. MEASURED-ELEVATION-POINT ELEVATION DIMENSION is in meters. An automated translation system would need to be implemented to achieve perfect alignment.

Appendix B.
A Critique of the OMSC Object Model

This appendix provides a brief critique of the design of objects in the OMSC Object Model. The OMSC Object Model is to serve as a framework for designing and developing simulation software. The developers hope that the model will lead to a repository of reusable software that can reduce the cost and time required to develop future simulation systems.

Achieving this laudable objective could have wide-ranging impact on Army programs. However, the model will yield good designs and reusable software if the objects in it are themselves well designed and implemented. A thorough critique of the model is therefore imperative.

The alignment analysis has uncovered several potential problems with the OMSC Object Model. Some of these problems are problems only insofar as alignment with C4I systems is concerned. The OMSC object model's developers did not consider alignment, so the OMSC obviously cannot be faulted for alignment flaws. These problems are mentioned for the sake of future M&S modeling efforts, whether the OMSC object model or some other model. Modelers will want to avoid design decisions that will hinder alignment.

Other problems in the OMSC object model are independent of alignment. The OMSC standard objects were developed by studying legacy systems, and the problems in this category may reflect flaws in the design of those systems. Avoiding these types of problems should promote ease of design, interoperability, and software reuse among M&S systems.

Briefly, the problems are as follows:

- The standard definitions have too many ambiguities to be useful.
- Some of the so-called objects are not really objects, but are instead encapsulations of related functions.
- The relationships between objects are not properly captured.

This appendix presents each problem, using examples drawn from the definitions of the Unit and Platform standard objects. It discusses why these problems are likely to impede the utility of OMSC objects. It suggests fixes for the problems.

This appendix concentrates on problems inherent in the high-level design decisions that went into creating the OMSC. It omits low-level details that relate to alignment. Section 7 gives these details.

It must be noted that the OMSC's creators regard the issues identified in this appendix not as problems but as appropriate design decisions in response to the OMSC's charter, which states that the standard objects, and the classes they contain, are to be simulation-independent. They took care to create "no specifications that would impact or direct implementation" [H 2000]. They believe that many of the recommendations in this appendix, as well as in Section 7, violate their charter.

Abstraction is a powerful tool for system design and reuse, but by definition an increase in abstraction yields some decrease in utility. The IDA study team believes that, if the OMSC is to promote interoperability and reuse, the standards it establishes must be more concrete than their current form. Time and experience will of course be the ultimate arbiters of whether the OMSC standard objects are appropriate.

B.1 Object Descriptions Have Ambiguities

The OMSC model is divided into a set of standard objects. Each standard object, such as Unit or Platform, is described in a single document. Each standard object is specified as a set of classes. One of these classes provides the definition for the object that is the focus of the document. The other classes are associated with the main object using inheritance or aggregation relationships. These relationships are shown in a picture.

Each class is specified using a textual definition and a set of public methods. Each public method is in turn specified using a textual definition that describes its effect. For example, the Unit class is defined using the text:

> **Class Unit:** A "Unit" is any military organization that is composed of multiple entities. Examples include military organizations such as a company, battalion, brigade, or division.

The Unit class has ten public methods, one of which is getVelocity(). The definition of getVelocity() is:

> getVelocity(): Returns the current velocity (direction of movement and rate) of the unit.

This is the extent of the specification. Details have deliberately been omitted, because the OMSC's designers feel that including them would constrain implementations. However, the missing details leave too much room for interpretation. An implementation is likely to contain objects that are application-specific. The following sections list ambiguities in the specifications of objects.

It is worth noting that many of the ambiguities could be eliminated by the adoption of more careful documentation standards. For example, following the style that Sun Micro-

systems uses to describe Java classes would of necessity clarify many of the issues described below.

B.1.1. Unknown Parameter Specifications

None of the methods list any parameters. For example, the Unit.move() method advances a unit towards its next location. The method's description does not indicate how a developer specifies the next location. Any of the following are possible:

move(x, y: float)	Coordinate pair, Cartesian model
move(x: latitude; y: longitude)	Coordinate pair, geodetic model
move(l: Location)	Coordinates using Location standard object
move(orientation: float, distance: float)	New location is relative to current location

These and other possible parameters (e.g., the amount of time taken for the move, or the route taken for the move) are left to the developer's imagination.

In a reuse library, the implementations of methods would of course have parameter schema specified. However, unless the designers of the OMSC act now to state their intent as to (for instance) how one specifies movement, a library resulting from the specifications is likely to consist of a collection of operations that apply only to the application for which they are developed.

B.1.2. Unknown Return Values

This issue is similar to unknown parameter specifications. The methods do not indicate the type of values they return. Often the reader cannot infer if a method returns a string or an object instance For instance, the Unit.getSide() method returns "the faction or coalition for the platform". This can be interpreted as either of the following:

- getSide(): String
- getSide(): Organization

There is an extra benefit to stating return types. It forces some thought about additional classes (such as Organization) that the OMSC might need. Consideration of these classes would increase the OMSC's completeness.

B.1.3. Unknown Units

The units in which numeric values are stated is never given. The story of the Mars Climate Orbiter, which crashed on the surface of Mars because an English/Metric inconsis-

tency put it in an orbit too close to the planet's surface, should underscore the importance of defining units up front.

B.1.4. Unknown Constructors and Modifiers

The descriptions never provide a class constructor, nor do they explicitly specify instance modifiers ("set*X*" methods). This means that the complete set of attributes expected of an instance for the purposes of simulation cannot be determined. The following constructor for Unit:

```
Unit(id: String, l: Location)
```

indicates that a unit must always have an ID and a location.

B.2 Class Design Is Not Based on Objects

The design of the model suggests that some classes were created to package related functions. There is nothing wrong with packaging functions into a class, but other parts of the model seem to imply that these packages are to be treated as collections of objects. An object, according to the usual definition, has a current state. A collection of functions does not have a state.

All the component classes of Unit (Communications, UnitGeometry, Intel, SystemGroup, Attrition, Logistics, and C2) are packages of related functions rather than classes of objects. The Platform class has two component classes, Sensor and Weapon, that from their descriptions seem like classes of objects. The other classes (Movement, Logistics, Crew, Communications, Carrier, and PlatformFrame) appear to be packages of related functions.

Aggregating related functions is a valid design practice. However, the practice is unlikely to yield significant software reuse when compared to a design based on objects. In any event, some analysis of the model reveals opportunities for an improved object-oriented design. The rest of this section makes some suggestions in that regard.

B.2.1. UnitGeometry

The UnitGeometry class has two methods: getShape() and getOrientation().

This class' name suggests that it models only the geometry of units. Other OMSC entities require geometry, too. For instance, platforms have a cross-sectional area signature that is used by sensors.

Therefore, it would be better to replace UnitGeometry with a class named Geometry, which can model the geometry of any entity. Geometry should be a virtual class. It should have subclasses for various types of shapes, such as (for two dimensions) rectangle and circle. See Figure B-1.

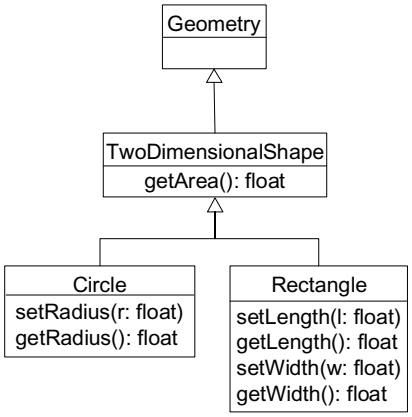


Figure B-1. Geometry Class Hierarchy

Note that this hierarchy does not have a getShape() or getOrientation() method. Geometry and its subclasses model the concept of geometry. The getShape() and getOrientation() methods belong in the Unit class, declared as follows:

- getShape(): Geometry
- getOrientation(): float

B.2.2. Communications

This class seems too atomic. The descriptions of the methods suggest the existence of Messages, Nodes, and Networks as primitive elements that make up the ability to communicate. The OMSC should have Message, Node, and Network classes.

B.2.3. SystemGroup

The methods of this class indicate that it is a placeholder for component systems or platforms associated with the unit. The methods can all be eliminated, and replaced by invocations of methods in the parent unit. That is,

```
u.sg.acceptGains(1)
```

is equivalent to:

```
u.addSystem(s)
```

This has the added advantage of explicitly associating a system instance with a unit, rather than just incrementing the number of systems of some type. Simulating the functionality of acceptGains() would of course present no challenge.

B.2.4. Attrition

The definition of the class reads as follows:

> The attrition component allows the unit to cause losses to another unit.

The capability of unit A to cause losses to unit B is not a component of the unit. It is a computable quantity based on unit A's current fire systems capability (or whatever destructive capability A might possess).

Modeling attrition therefore requires knowing how much destructive capability one unit aims at another. It is this factor that should be encapsulated. The causeAttrition() function should be packaged but need not otherwise be part of a class associated with Unit.

B.2.5. Logistics

The description of the Logistics class indicates that it is:

> intended to capture or represent the internal logistics capability and/or requirements of the unit.

This description is at odds with Logistics.receive(), which is "used to increment the quantity of this logistic component." Incrementing a quantity does not capture capability.

In any event, capturing requirements does not imply the need for an object instance. It rather describes a set of attributes that could just as easily be part of the Unit class.

As for incrementing (or decrementing) quantity of a logistics component, the component can be part of a Unit. See Section B.2.3.

B.3 Inter-Class Relationships Do Not Follow a Consistent Model

The OMSC Platform and Unit documents show a design of each object as a picture. This picture contains classes, and two types of inter-class relationships: inheritance and aggregation.

The inheritance relationships are not stated according to usual modeling conventions. They show a label of "0+" (figures in this document are modifications of those in OMSC documents). This label is meaningful in an aggregation relationship, but not in an inheritance hierarchy. The labels should be removed from inheritance relationships.

The aggregation relationships all have "0+" labels. In an object model, this label indicates a relationship between a parent and a child; it means that zero or more instances of the child are associated with the parent.

However, because most of the classes other than Unit and Platform are packages of functions rather than collections of objects, a 0+ relationship has no meaning. For example,

the Platform object has a 0+ relationship with the Movement object. Movement is a collection of 3 functions (getVelocity(), changeVelocity(), and moveTo()) with no obvious underlying object being part of the class. Obviously the relationship does not mean that there are different instances of each function that might be invoked. How, then, should it be interpreted?

Most likely, the relationship indicates packaging and not aggregation. The diagram notation needs to be extended to capture this new relationship.

Appendix C. AICDM Views and Their Status

This appendix lists the views in the current version of the AICDM. These views give a flavor of the AICDM's intent: to model battlefield objects. The AICDM attempts to model anything that might be relevant to the warfighter. Personnel, materiel, organizations, terrain, and organizations are all within its purview.

Table C-1 lists the views, in alphabetical order. The following is an explanation of the entries in the Status column:

- To be worked: It is recognized that work needs to be performed on the view. However, there is no timetable for performing the work.
- Working: It is recognized that work needs to be performed on the view. Work is ongoing, and there is a timetable for performing the work.
- Package: Submitting a view for changes and revisions in accordance with DoD-8320.1-M-1 requires preparing a data proposal package containing, among other things, an ERWin model of the data together with metadata specifications. This status means the package has been created and is ready to be released for cross-functional review and eventual approval.
- To be Updated: After a data proposal package has been submitted, experts generally make comments. If the resolution of these comments requires further changes to a view, that view's status is "to be updated" until the changes are made.
- Approved: The updates are finished.

Table C-1. AICDM Views and Their Status

View	Status
ACTION	Approved
Action-Resource-Employment View	To be worked
CANDIDATE-TARGET	Approved
CAPABILITY	Approved
DETAILS Task Action View	To be worked
DETAILS skill/pos/occ-2 View	To be worked
Enemy Org View	To be worked
Establishment	Package
Feature View	Approved
Force Package Type view	To be worked

View	Status
Graphic	Package
Holding View	Package
Information Reference	Package
LOCATION	Working
MATERIEL item and type View	To be worked
ORGANIZATION	Approved
ORGANIZATION-TYPE	Approved
Operational Specialty View	To be worked
Plan View	Approved
Task View	To be worked
Version 1 Full Model	To be worked
mat-item View	To be worked
met View	To be worked
met-feature View	To be worked
skill/pos/occ View	To be worked
standards compliant full model view	To be up-dated
ujtl View	To be worked

Appendix D. Specific Recommended Changes

This appendix splits the specific recommendations from Section 7 into separate tables, one for each OMSC standard object. The intent is to present recommendations categorized by OMSC classes and methods.

D.1 Unit Standard Object

Unit Class

The getLocation() Method	
Issues	• The OMSC is ambiguous about how a unit location is modeled. It says: Typically [current unit location] is the center of mass or some other point location representative of the unit location. If center of mass is typical, there must be some atypical representations. The OMSC does not enumerate them. • The OMSC does not define how center of mass is computed. M&S applications use different models that depend upon factors such as level of aggregation (corps/division vs. theater) and whether the simulation is for training or analytical purposes. (In some M&S training applications, center of mass is not computed at all, but is estimated by (and entered by) a human operator.)
Recommended Changes to the OMSC	• The OMSC should enumerate all valid ways in which a unit's location can be modeled. A better approach would be to accept center of mass as the standard way to represent location. Other location-like quantities (e.g., areas) should be represented using other classes. • The Location class should have a virtual method centerOfMass(). An M&S application based on the OMSC would need to supply a definition for the method.

The getVelocity() Method	
Issues	• The AICDM does not model velocity. • The OMSC's description does not prescribe the units in which velocity components are expressed.
Recommended Changes to the AICDM	The AICDM needs to be able to model the velocity of (at least) the PERSON, MATERIEL, and ORGANIZATION entities.
Recommended Changes to the OMSC	The OMSC should define a (parameterized) model of velocity. The model must describe units and degree of precision (or these quantities must be parameters).

The getID() Method	
Issues	The OMSC does not indicate how the ID might be used. Use can prescribe format. (E.g., if an ID labels units on a screen, it must be short.) This influences whether the AICDM can model OMSC IDs using existing attributes.
Recommended Changes to the OMSC	The OMSC should define a model of IDs based on their use in existing M&S systems. Ideally, the OMSC should use C4I IDs.
The getSide() Method	
Issues	• The OMSC does not indicate the maximum number of sides that must be supported. At one time it was assumed that the only sides would be "blue" and "red", but with the advent of coalition warfare the number of possible sides has grown. Note that WARSIM has a contractual requirement to support at least 36 different sides. • The OMSC states that there is no implied enmity between sides, but does not state the possible relationships between sides. • The AICDM does not have an explicit model of sides, although one can be simulated using ORGANIZATION structures.
Recommended Changes to the AICDM	The AICDM should enhance its ability to model friends and foes. One approach is to add values FACTION and COALITION to the IDENTIFICATION-FRIEND-FOE CODE attribute of the ORGANIZATION entity. Another is to externalize the FRIEND-FOE attributes.
Recommended Changes to the OMSC	• The OMSC should state explicitly whether there is an implied maximum number of sides. • The OMSC should enumerate the possible relationships between sides.
The getPosture() Method	
Issues	• The OMSC does not enumerate all possible postures. • The OMSC does not relate postures to other objects. For instance, a posture such as "hasty defense" implies that a unit's velocity is zero. • The AICDM has a limited set of attributes that can represent posture. Their values do not cover the examples of status given in the OMSC.
Recommended Changes to the AICDM	The AICDM needs to add an attribute associated with an ORGANIZATION that can represent posture. Since different types of organizations can have different types of posture (e.g., a civilian organization will not have the same types of a posture as a military organization), it might make sense to have multiple posture attributes, each associated with a subtype of ORGANIZATION.
Recommended Changes to the OMSC	• The OMSC needs to clarify the role that postures play in simulations and the possible values they may have. Do they need to be distinguished from current activities, as elaborated by the current activity codes of the ORGANIZATION and MILITARY-UNIT entities? • Whatever a posture may be, the set of valid postures for a given M&S system can apparently be expressed as an enumeration. This suggests what the result of *u*.getPosture() should be.
The getStatus() Method	

Issues	• The OMSC does not enumerate the different types of status. • The AICDM has a very limited set of attributes that can represent status. Their values do not cover the examples of status given in the OMSC.
Recommended Changes to the AICDM	Necessary changes to the AICDM cannot be determined until the concept of status in M&S systems is better understood. However, it is likely that: • The AICDM will need to accommodate numeric descriptions of status (e.g., as percent effectiveness). • The AICDM will need additional codes.
Recommended Changes to the OMSC	The OMSC should define a virtual class *Status*. Subclasses of *Status* would exist for every entity that needs to record status. Subclass methods would record and provide different facets of status as appropriate to the entity.
The getMission() Method	
Issues	The OMSC gives no structure to a mission or task. A mission is only free text. The AICDM, by contrast, has a rich structure that the OMSC cannot model.
Recommended Changes to the OMSC	the OMSC should define a Mission standard object that provides a more precise specification of what tasks are and how they will be used.
The move() Method	

Issues	• The OMSC does not state the parameters of the method. Potential parameters include:
	  • New coordinates (either relative or absolute)
	  • Time until new coordinates are reached
	  • The specification does not state whether the next location is known in advance (in which case parameters might not be needed).
	• The OMSC does not describe necessary granularity of movement (that is, minimum length or time of movement).

Recommended Changes to the OMSC	The OMSC should included a parameterized model of movement, one that addresses granularity.
The determineAttrition() Method	
Issues	The OMSC does not specify units for attrition (percentage? absolute values? specific entities removed?)
Recommended Changes to the OMSC	The OMSC should specify ways in which attrition may be stated. Where applicable, the OMSC should specify units and precision for attrition.

The UnitGeometry Class

The getShape() Method	
Issues	The OMSC does not indicate the nature or generality of bounding shapes.
Recommended Changes to the OMSC	The OMSC should define the result of getShape() to be a class *Shape*. *Shape* should have subclasses such as *Rectangle*, *Circle*, etc. This structure will support current needs while providing for future enhancements.
The getOrientation() Method	
Issues	• The OMSC does not state the units of orientation. Presumably orientation is in degrees. • The AICDM does not model orientation.

Recommended Changes to the AICDM	The AICDM should include information on orientation. It is likely that orientation must be specified for several types of entities (organizations and platforms, at least). Given the attributes of POINT, which include precision, it is probably necessary to create a new entity, ORIENTATION, containing attributes that express orientation. Instances of this entity would be linked to an ORGANIZATION (or PLATFORM) through a many-to-many relationship via an intermediate ORGANIZATION-ORIENTATION entity (similar to the relationship between ORGANIZATION and LOCATION). An orientation may be absolute or relative (e.g., "in front of"), and the AICDM should be able to model both types.
Recommended Changes to the OMSC	The OMSC should state the units and precision of orientation.

The Platform Class

This class is treated separately. See Section D.2.

The PlatformInfo Class

This class is treated separately. See Section D.2.

The Supply Class

The getRemainingCapacity(), getTotalCapacity(), and transfer() Methods	
Issues	• The OMSC does not prescribe how capacity is expressed. • The AICDM does not model capacity.
Recommended Changes to the AICDM	The AICDM needs entities that model capacity.
Recommended Changes to the OMSC	The OMSC needs a standard model, or perhaps set of models, for expressing capacity.

The Maintenance Class

The conductMaintenance() Method	
Issues	• The OMSC does not specify how maintenance actions or medical treatment are stated. It does not specify resultant states. • The OMSC does not specify if maintenance is performed on individual items or on groups of items. • The AICDM does not have an attribute of ACTION that can record maintenance.
Recommended Changes to the AICDM	A new value, CONDUCT MAINTENANCE, needs to be added to the ACTION-VERB CODE attribute of ACTION.
Recommended Changes to the OMSC	The OMSC needs a more exact model of maintenance. It must be possible to specify the type of maintenance to be performed, the expected result, the materials and effort consumed during maintenance, etc.
The conductRecovery() Method	
Issues	The OMSC describes recovery as if it were an all-or-nothing action: recover all entities. Presumably an M&S system can be more selective, opting to perform triage (for example).

Recommended Changes to the OMSC	The OMSC needs a model of recovery that describes: • How to determine what to recover. • The resources consumed during recovery.
The conductEvacuation() Method	
Issues	• The OMSC does not state how the location of a rear area is determined. • The OMSC does not define if evacuation must be performed en masse. Must a whole unit be evacuated, or can portions be evacuated?
Recommended Changes to the OMSC	• The OMSC should include a location as a parameter to the method. • The OMSC needs a model of evacuation that, like recovery, precisely defines what recovery is and the resources it consumes.

The C2 Class

The doC2() Method	
Issues	The OMSC does not describe the nature of command decisions or control actions.
Recommended Changes to the OMSC	The OMSC should create a set of classes that can model C^2 actions important to for M&S.

The Attrition Class

The causeAttrition() Method	
Issues	• The OMSC does not define the actions one unit can take that might cause losses to another unit. If those actions include firing weapons, it would seem more logical to invoke the weapon firing method directly than to invoke it through this method. • The OMSC does not specify how much attrition is caused by invoking this method.
Recommended Changes to the OMSC	The OMSC needs a model that relates attrition to the various actions one unit might take against another.

The Communications Class

The getNet() and setNet() Methods	
Issues	• The OMSC does not specify the types of objects that might be modeled in a network. Presumably it includes components of the unit hierarchy. It may also include platforms. • Does “Capable of exchanging messages” include both friends and foes? • Is the intent of communications to capture electronic message exchange? Or does it include other types of signals (e.g., semaphores, written communication, or for that matter verbal communication)?
Recommended Changes to the OMSC	The OMSC needs a more precise model that captures how units communicate in simulations.

The Intel Class

The collect() Method	
Issues	• The OMSC does not specify how the organic sensor assets of a unit are defined. • Detection involves more than turning on a sensor. The sensor is typically directed against some feature, other organization, etc. The OMSC does not define how search capabilities may be effected other than turning them on. • The AICDM does not model sensor activation and deactivation.
Recommended Changes to the AICDM	The AICDM needs attributes that give the state of a sensor, as well as other possible sensor-specific characteristics (orientation, e.g.)
Recommended Changes to the OMSC	• The OMSC must specify how the sensor assets of a unit are defined. • The OMSC must specify how the sensor assets of a unit may be used.
The reportContacts() Method	
Issues	• The OMSC does not specify the nature of results, nor how they might be used. The only other method associated with a Unit that might use intelligence is C2.doC2(). • Typically, M&S systems can model any entity detected by intelligence. The AICDM can only record such an entity if it is a candidate target.
Recommended Changes to the AICDM	• The AICDM needs to model entities recorded by intelligence but not yet known to be (candidate) targets. • The codes associated with FEATURE need to account for intelligence. • The proposed AICDM entity INFORMATION-REFERENCE may be useful for modeling intelligence results. The domain values of the INFORMATION-REFERENCE-CATEGORY-CODE attribute would need to be extended to account for intelligence.
Recommended Changes to the OMSC	The OMSC needs a class that acts as a superclass of possible results for the collect() method.

D.2 Platform Standard Object

The Platform Class

The getType() Method	
Issues	The OMSC does not define what a type designation is. Assuming that Platform is a virtual class and that actual platform instances would be members of subtypes of Platform, the type designation is probably just an identifying string that gives the name of the subtype.
Recommended Changes to the AICDM	For convenience, the AICDM may need to be amended to include a new entity class PLATFORM. Subtypes of this entity would include all AICDM entities that are considered platforms in the OMSC. (The DDA contains separate entities for ships and satellites. We recommend including these entities in the AICDM to address M&S requirements for water and space platforms.)

Recommended Changes to the OMSC	The OMSC needs to enumerate at least a partial list of valid platform types.
The getLocation() Method	
See the description of the getLocation() method for the Unit class.	
The getSide() Method	
See the description of the getSide() method for the Unit class.	
The assessDamage() Method	
Issues	• The OMSC does not specify the types of objects that might cause damage. • The OMSC does not specify how the amount of damage is determined. • The description states that the method calculates damage caused, but there is no method associated with a Platform that actually causes damage. • The OMSC does not specify units in which damage is measured. • The OMSC does not specify how the object that caused the damage is identified, if more than one object can cause damage, or what an object might be (can damage to a truck come from a rock? If not, how does one model such potential damage?). • This method assesses damage. How is damage caused? What entities are consumed in causing it? • The AICDM codes for damage to materiel, facilities, and people are not adequate to model the types of damage possible in an M&S system.
Recommended Changes to the AICDM	The AICDM needs codes for materiel, facilities, and people that model the types of damage possible to such entities in M&S systems. Possibly some types of damage are complex enough to warrant creation of a new entity, with attributes to model the various types of damage.
Recommended Changes to the OMSC	The OMSC needs a model of damage.

The Sensor Class

The getMaxRange() Method	
Issues	The OMSC does not prescribe the units of range.
Recommended Changes to the OMSC	The OMSC should standardize the units and precision of range.
The getOrientation() Method	
See the description of the getOrientation() method for the UnitGeometry class.	
The getContacts() Method	
Issues	• A Platform does not have a location. Without a location, how can the contacts be determined? (The Unit class has a collection of Platform instances, and moreover may be hierarchically composed of Unit instances. Perhaps a Platform's location is always the center of mass of its containing Unit, where the Unit's granularity is small enough to resolve the problem of locating contacts.)
Recommended Changes to the OMSC	The OMSC needs to clarify the method by which contacts are gathered, and whether it requires a location; if it does, the OMSC needs to clarify how the location of a platform is determined.

The activate() and deactivate() Methods	
Issues	The AICDM does not model whether a sensor is on or off.
Recommended Changes to the AICDM	The AICDM needs to model sensor state.

The Weapon Class

The getMaxRange() Method	
Issues	The OMSC does not prescribe the units of range.
Recommended Changes to the OMSC	The OMSC needs to define the units and precision of range.
The load() Method	
Issues	• The OMSC's description of loading "a" munition seems oriented towards particular types of weapons, like artillery. Is that the intent? • The AICDM does not model whether or not a weapon is loaded.
Recommended Changes to the AICDM	The AICDM needs to be able to model the state of a weapon.
Recommended Changes to the OMSC	The OMSC needs to clarify the range of weapons to which this operation can be applied.
The engageTarget() Method	
Issues	• The OMSC does not indicate how long the weapon-firing event takes. Is this captured in a subtype? It should be a virtual method. • There is no method to determine whether weapon firing has ended. • Is aiming part of engaging a target? There is no aim() method. • The AICDM does not model weapon firing.
Recommended Changes to the AICDM	The AICDM needs to model weapon firing. More precisely, it needs to model the status of a weapon, including being in a firing state (whatever that might mean for a particular weapon).
Recommended Changes to the OMSC	The OMSC needs a model of weapon firing. The model should allow a system to determine if a weapon has been fired, where it's pointing, and things like that.

The PlatformFrame Class

The getSignature() Method	
Issues	• The OMSC lists examples of, but does not describe the full range of, signatures. Nor does it provide a general means to encapsulate signatures (e.g., a virtual class Signature). • The method's description is worded as if a target has a single signature. Can't a target have multiple signatures? • The AICDM can model only a cross-sectional area signature.
Recommended Changes to the AICDM	The AICDM should increase the range of signatures that it can model. (The DDA has attributes to keep track of height, weight, etc. that the AICDM lacks.)
Recommended Changes to the OMSC	The OMSC should define a virtual class *Signature*, which would be the value returned by getSignature(). Sensors would return subclasses (ThermalSignature, AcousticSignature, etc.).

The Movement Class
The getVelocity(),changeVelocity(), and moveTo() Methods
See the description of the getVelocity() method for the Unit class.
The Logistics, Supply, Maintenance, and Communications Classes
See the description of these classes for the Unit standard object.

D.3 Location Standard Object

The issues surrounding locations have been discussed in Section D.1.

Appendix E. Implementation of Recommendations in Later Versions of the AICDM

As noted in the main portion of this report, IDA performed the alignment analysis using the March 1, 2000 version of the AICDM. The AICDM, however, is meant to continue evolving to encompass any new valid data requirement. As a result, some of the recommendations made during the preparation of this report have been already incorporated into the AICDM, and data package proposals for submission to the DoD-8320 standardization process have been prepared. This Appendix shows the AICDM's status as of January 2001.

E.1. Velocity and Orientation Attributes

This report has made two recommendations regarding the need for the AICDM to have attributes that can model velocity and orientation:

Section 7.1.1: The AICDM needs to be able to model the velocity of (at least) the PERSON, MATERIEL, and ORGANIZATION entities.

Section D.1: The AICDM should include information on orientation. It is likely that orientation must be specified for several types of entities (organizations and platforms, at least). Given the attributes of POINT, which include precision, it is probably necessary to create a new entity, ORIENTATION, containing attributes that express orientation. Instances of this entity would be linked to an ORGANIZATION (or PLATFORM) through a many-to-many relationship via an intermediate ORGANIZATION-ORIENTATION entity (similar to the relationship between ORGANIZATION and LOCATION). An orientation may be absolute or relative (e.g., "in front of"), and the AICDM should be able to model both types.

The January 2001 version of the AICDM includes several attributes to model velocity and orientation. See Table E-1.

Table E-1. New AICDM Attributes to Model Velocity and Orientation

Entity	New Attributes
FACILITY[12]	FACILITY-POINT BEARING ANGLE FACILITY-POINT SPEED RATE

[12] The analysis in Appendix A uses a FACILITY to model a stationary entity. However, the Navy proposes to model a ship as a FACILITY.

ORGANIZATION	ORGANIZATION-POINT BEARING ANGLE ORGANIZATION-POINT SPEED RATE
MATERIEL	MATERIEL-POINT BEARING ANGLE MATERIEL-POINT SPEED RATE

Figure E-1 shows these attributes, the entities containing them, and selected relationships of these entities to key AICDM entities.

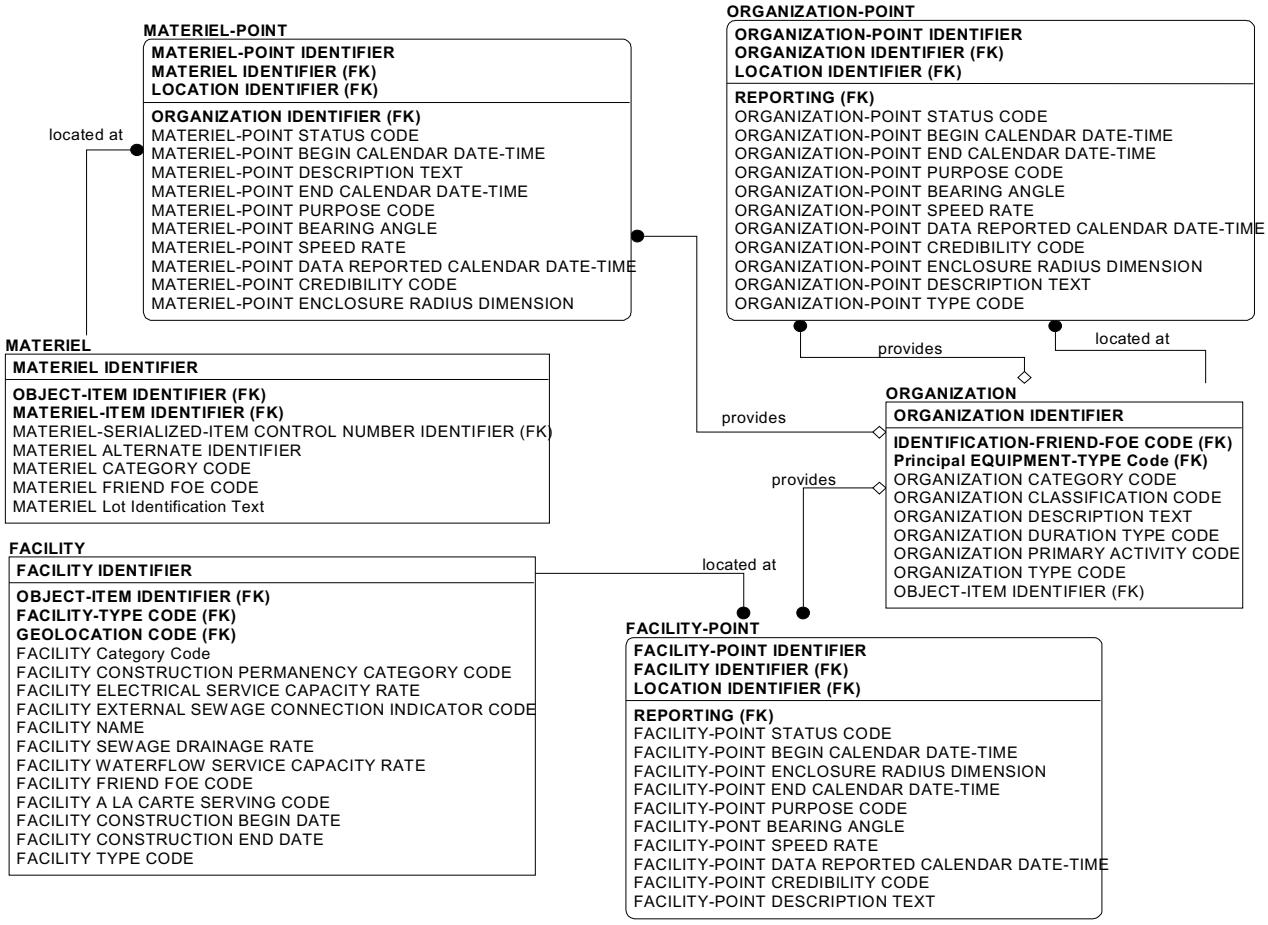


Figure E-1. Velocity and Bearing Angle Attributes for FACILITY, MATERIEL, and ORGANIZATION in the January 2001 version of the AICDM

This data proposal package is still undergoing technical review. A request to model PERSON-POINT in the same way as the three entities mentioned above has already been submitted. Furthermore, analysis with respect to maritime operations appears to warrant extending the same approach to FEATURE-POINT.[13]

[13] The March 2000 version of the AICDM modeled an organization's center of mass as a LOCATION, one subtype of which was POINT. The January 2001 version of the AICDM relates a battlefield object's center of mass directly to a POINT.

E.2. Externalization of IDENTIFICATION-FRIEND-FOE

This report has made a recommendation regarding the AICDM's ability to model coalition forces:

Section 7.1.1: The AICDM should enhance its ability to model friends and foes. One approach is to add values FACTION and COALITION to the IDENTIFICATION-FRIEND-FOE CODE attribute of the ORGANIZATION entity. Another is to externalize the FRIEND-FOE attributes.

The January 2001 version of the AICDM has already replaced the table attributes for IDENTIFCATION-FRIEND-FOE that existed in ORGANIZATION and FEATURE by a foreign key coming out of the externalized entity IDENTIFICATION-FRIEND-FOE. See Figure E-2. Similar replacements will occur as the packages for the MATERIEL, PERSON and FACILITY domains are prepared.

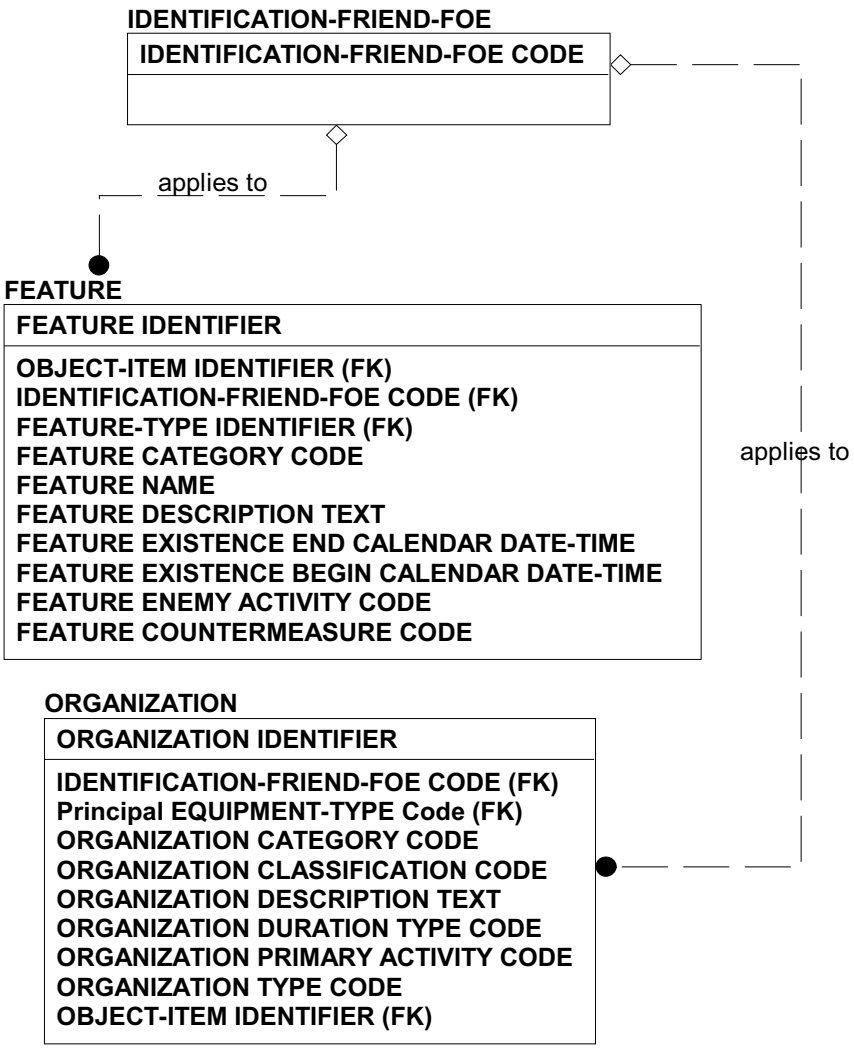


Figure E-2. Externalization of IDENFICATION-FRIEND-FOE in the latest version of the AICDM

The current enumerated domain values for IDENTIFCATION-FRIEND-FOE CODE are: ASSUMED FRIEND, FRIEND, HOSTILE, JOKER, FAKER, NEUTRAL, PENDING, and SUSPECT. Review of the proposed values FACTION and COALITION may be necessary.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2001	Final

4. TITLE AND SUBTITLE
Alignment of Army Integrated Core Data Model and Object Management Standards Category

5. FUNDING NO.S
DASW01-98-C-0067
Task Order BC-5-1943

6. AUTHOR(S)
Brian A. Haugh, Task Leader, Steven P. Wartik, Francisco L. Loazia, and Michael R. Hieb

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NO.
IDA Paper P-3596

SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of the Secretary of the Army
ODISC4 (SAIS-PAA-S)
2461 Eisenhower Avenue, Room 1126
Alexandria, VA 22331-0700

10. SPONSORING/MONITORING AGENCY REPORT NO.

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution: 11 February 2002.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

Interoperability between Command, Control, Communications, Computers, and Intelligence (C4I) and Modeling & Simulation (M&S) systems is an unrealized goal with great potential. Interoperability can facilitate training, simulation-based acquisition, mission planning and rehearsal, and course of action development and analysis. Recent interoperability research has concentrated on general-purpose approaches that can provide standards and reuse across a wide range of systems. One important component of interoperability is data model alignment, the degree to which the data models of two systems use the same elements. This paper presents a rigorous definition of data model alignment and uses it to assess the degree of alignment between the Army Integrated Core Data Model (AICDM) and the standard objects defined by the Object Management Standards Category (OMSC), two important emerging standards in the C41 and M&S communities, respectively. This assessment is used to make recommendations on changes to each model that would promote interoperability. The conclusion is that the OMSC standard objects need considerable work to model C4I data.

14. SUBJECT TERMS
Army Integrated Core Data Model (AICDM), Object Management Standards Category (OMSC), C4I, Interoperability, Data Model Alignment, Object Model, Modeling and Simulation

15. NO. OF PAGES
192

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL